A Hybrid Approach to the Sentiment Analysis Problem at the Sentence Level

Orestes Appel

Submitted in partial fulfilment of the
requirements for the degree of Doctor of Philosophy
at De Montfort University

Leicester, Great Britain
July, 2017.

Acknowledgements

I would like to thank my wife Gina and our children Arianna & Jane and Jan Alexander & Michela -the driving force- and our animal kingdom friends, Luna, Bruno, Freyja, Kovu, Enzo and Bamm-Bamm, for their love, encouragement, support, patience, imagination and interest. I would also like to thank my supervisors Prof. Francisco Chiclana, Dr. Jenny Carter, Prof. Hamido Fujita and Prof. Halia Valladares-Montemayor for their support and invaluable guidance.

In addition I would like to thank:

- Mamá y Papá.
- One: you changed my life!
- Arianna & Jan for their quickness and depth; but most importantly, thank you for being my children! The honour is all mine!
- In memoriam of Mamina, Orestes Appel Urdaneta, Olga Maduro Guadarrama, Nana, María Barnabina Santucci, Mariolga, Antonio Pavone, T-Rex, Jan Appel Urdaneta, Estrella Maduro Guadarrama, and all those that are not here any more, but are always in my heart.
- My closest friends: Rafael Baralt, Eduardo Fleischer, Federico Leáñez, Alejandro Martínez, Gerardo Pacanins, Angel Puerta, José Gregorio Silva and especially, Gustavo Núñez Testa [1958 - 2013] .
- Prof. Carlos Di Prisco, who does not even know that his seminar in *Logic and Gödel* at Universidad Central de Venezuela, sparked my interest in Mathematics & Computer Science.
- Special personal thanks to those artists, authors, philosophers and scientists that indirectly guided my steps by enriching my life:
    - Jorge Luis Borges [1899 - 1986]: by reading his opus I had my very personal World-Wide Web back in the 1970s; with every page and reference, an opportunity to search for more, and more and more!
    - Lawrence Durrell, for The Alexandria Quartet.
    - W. Somerset Maugham & Herman Hesse, for Larry, Lara, Demian and Siddhartha.
    - J.R.R. Tolkien, pues no todos aquellos que andan errantes están perdidos.
    - Ingmar Bergman, for The Seventh Seal (Swedish: Det sjunde inseglet) [1957].
    - Bach, Handel, Mahler, Mozart & Puccini.
    - ELP, Peter Gabriel, Genesis, King Crimson & Yes.
    - Bertrand Russell, Alan Turing (do we need to ask why?), Kurt Gödel, John McCarthy, Alain Colmerauer, Robert Kowalski, Guy L. Steele, Gerald Jay Sussman and Edsger Dijkstra.

Abstract

This doctoral thesis deals with a number of challenges related to investigating and devising solutions to the Sentiment Analysis Problem, a subset of the discipline known as Natural Language Processing (NLP), following a path that differs from the most common approaches currently in-use. The majority of the research and applications building in Sentiment Analysis (SA) / Opinion Mining (OM) have been conducted and developed using Supervised Machine Learning techniques. It is our intention to prove that a hybrid approach merging *fuzzy sets*, a *solid sentiment lexicon*, *traditional NLP techniques* and *aggregation methods* will have the effect of compounding the power of all the positive aspects of these tools.

In this thesis we will prove three main aspects, namely:

1. That a Hybrid Classification Model based on the techniques mentioned in the previous paragraphs will be capable of:

   (a) performing same or better than established Supervised Machine Learning techniques -namely, Naïve Bayes and Maximum Entropy (ME)- when the latter are utilised respectively as the only classification methods being applied, when calculating *subjectivity polarity*, and

   (b) computing the *intensity of the polarity* previously estimated.

2. That cross-ratio uninorms can be used to effectively fuse the classification outputs of several algorithms producing a compensatory effect.

3. That the Induced Ordered Weighted Averaging (IOWA) operator is a very good choice to model the opinion of the majority (consensus) when the outputs of a number of classification methods are combined together.

For academic and experimental purposes we have built the proposed methods and associated prototypes in an iterative fashion:

- Step 1: we start with the so-called Hybrid Standard Classification (**HSC**) method, responsible for subjectivity polarity determination.
- Step 2: then, we have continued with the Hybrid Advanced Classification (**HAC**) method that computes the polarity intensity of opinions/sentiments.
- Step 3: in closing, we present two methods that produce a semantic-specific aggregation of two or more classification methods, as a complement to the HSC/HAC methods when the latter cannot generate a classification value or when we are looking for an aggregation that implies *consensus*, respectively:
  - the Hybrid Advanced Classification with Aggregation by Cross-ratio Uninorm (**HACACU**) method.
  - the Hybrid Advanced Classification with Aggregation by Consensus (**HACACO**) method.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

BoW Bag of Words

FIS Fuzzy Inferencing System

FS Fuzzy Sets

FL Fuzzy Logic

HSC Hybrid Standard Classification

HAC Hybrid Advanced Classification

HACA Hybrid Advanced Classification with Aggregation

IOWA Induced Ordered Weighted Averaging

LBM Lexicon-Based Method(s)

LSA Latent Semantic Analysis

ME Maximum Entropy Method

MF Membership Function

ML Machine Learning

MLA Machine Learning Approach

NB Naïve Bayes Method

NLP Natural Language Processing

NLU Natural Language Understanding

OCC Ortony-Clore-Collins Model

OL Opinion Lexicon

OM Opinion Mining

OWA Ordered Weighted Averaging

PMI Pointwise Mutual Information

PMI-IR Pointwise Mutual Information-Information Retrieval

POS/PoS Part Of Speech

SA Sentiment Analysis

SL Sentiment Lexicon

SO Semantic Orientation

SOTA State Of The Art

SML Supervised Machine Learning

SR Semantic Rules

SVM Support Vector Machine

TM Text Manipulation

UML Unsupervised Machine Learning

VSM Vector Space Model

VSMs Vector Space Models of Semantics

Part I

INTRODUCTION & BACKGROUND

Chapter 1

Report Organisation

> They say that during a dark night it is not possible to know where to go without technology; but, what if we happen to enjoy the benefits of a starry night?
>
> Unknown

Before we start to get into the body of work we will be presenting in this report, we thought that it was adequate to show a high altitude view of the organisation of this document. Hence, we will mention the parts that make this PhD thesis report and provide some information on what each of them encompasses. The structure of this report is as follows:

Part I - *Introduction & Background*: Part I supplies a gentle introduction to the topic of Sentiment Analysis (SA) and Opinion Mining (OM), in order to provide context for our research. It includes a literature review of the SA discipline, that in turn contributed to the effort presented in article [10].

Part II - *Motivation, Hypothesis & Methodology*: Part II describes the motivation behind our research as well as our initial research hypothesis, which became the drivers behind the work we have done and that has been documented in this report. We explain as well the research approach that was followed providing details on how the experiments were conducted, what data sets were used and how the results were analysed.

Part III - *Methods, Techniques & Tools*: The main methods, techniques and tools utilised in the sentiment analysis discipline and research area are addressed in Part III. We present the two main approaches that are current: the lexicon-based approach and the machine learning approach. Coverage related to other techniques and tools that we have utilised in our research, especially on the *aggregation* side, are included as well in this part. We have included the main techniques and elements and have provided plenty of references for the curious readers looking for more details. The specifics of how some of the techniques described here will be utilised, as well as some others tools to be introduced in the Proposed Solution chapters, will be addressed in Part IV. Having said that, there will be some inevitable overlapping between Part III and Part IV, with Part III presenting generic ideas about a given topic, method or technique, and Part IV covering the same topic in more details and discussing as well specific implementation information. Material from this Part III contributed to articles that have been published already by the author of this report. See references [13, 17, 18].

Part IV - *Proposed Solution & Experimental Results*: The proposed solutions we have devised as a consequence of this research effort is presented in Part IV, along with the results associated with the experiments that have been conducted. As we have taken an incremental approach to building the proposed

solution(s), the actual methods are presented accompanied by their respective experimental results. Especially, because the material here developed became central to a number of articles published by the author in journals and conferences' proceedings. Significant segments of material presented in this Part IV, contributed to articles where the author of this report was the main contributor. See references [10, 11, 12, 13, 14, 15, 16, 17, 18]. Special attention deserve Chapters 14, 15 and 16, where the match with three published journal papers is very high. *We have proposed one main method and two other ones that can be seen as enhancements to the first one. These three methods that embody the most significant contributions we have made to the body of knowledge of the Sentiment Analysis discipline, were shared with the research community through the following journal articles before we completed this thesis' report*. The devised methods and associated articles are the following:

1. A Hybrid Approach to the SA Problem at the sentence level, presented in reference [13] and covered in Chapter 14 of this report.
2. An enhancement to the method described in (1) above, founded on *Aggregation by Cross-ratio Uninorm*, shared in reference [18] and presented in Chapter 15 of this document.
3. An extension to the method described in (1) above, based on *Aggregation by Consensus*, discussed in reference [17] and fully described in Chapter 16.

Part V - *Conclusions & Further Work*: Part V includes our conclusions and some recommendations related to possible further work. Nevertheless, keep in mind that each of chapters 14, 15 and 16, are self-contained in terms of the description of the proposed solution, its experimental results and their associated conclusions & further work. As we have ended up devising one common solution on top of which three other methods have been built, Part V summarised the findings already discussed in Part IV. Material from this Part V contributed to articles that have been published already by the author. See references [10, 11, 12, 13, 14, 15, 16, 17, 18].

Part VI - *Appendices*: We close this thesis' report with the Appendix part, including full references to work published by the author, Scheme & Python code samples from the proof-of-concept prototypes, data preparation processes and code utilised and examples of the outputs of the different classification methods.

Chapter 2

Sentiment Analysis Main Concepts

> Without an alphabet, abstraction becomes an exercise in futility.
>
> Unknown.

A significant part of the content of this chapter was used in an article published by the author in 2015. See reference [10]. In this chapter we will discuss the main components of Sentiment Analysis (SA) -or Opinion Analysis- as a discipline. It will go deeper on what the main challenges are and it will stress topics that we would like to concentrate as areas of research.

2.1 Key Ideas

SA is a discipline that has seen a lot of activity since approximately the year 2000 [133]. The main reason for that, so it seems, is the proliferation of social media and its tools (i.e. Tweeter, Facebook, LikedIn, etc.), that has made the availability of information about *how people feel about things* more readily available to the masses. In addition, companies and other profit and non-profit organisations have accumulated a vast amount of data as well on how their employees or customers *feel* about the products and services they receive from the aforementioned organisations. Even Human Resources divisions, are keen on understanding whether a potential employee will be loyal and become a long-term member of the company or she would leave after receiving training and benefits.

In a way, a discipline that started as a research topic in Natural Language Processing (NLP) in Computer Science schools around the world, has now made a transition to other departments in academia and the industry, like those more related to business and management schools. The reason is very simple: everyone wants to maximise their profits, and getting to understand what people think about oneself and one's company could make a big difference business-wise.

According to some respected researchers [133, 139], there are many challenges lying ahead for SA. Many are the reasons, but the fact that NLP has been around for a long time but only has focused on Opinion Analysis recently, suggests the intrinsic difficulties with this discipline.

Sentiment Analysis combines in its very own way the application of NLP, Computational Linguistics and Text Analysis. A definition attributed to Michelle de Haaff, in her article "Sentiment Analysis, Hard But Worth It", published in her *blog* appeared at the web page CustomerThink (2010), is as follows: "classifying the polarity of a given text at the document, sentence, or feature or aspect level, whether the expressed opinion in a document, a sentence or an entity feature or aspect is positive, negative, or neutral". She goes ahead as well to loosely define Advanced Sentiment Analysis, like the one that goes 'beyond polarity' sentiment classification, and it looks, for instance, at emotional states such as 'angry', 'sad', and 'happy'.

Understanding the emotions being conveyed by a given source, may it be a tweet, a document, a report, a blog, a segment of a politician speech, etc., has proven to be an important activity for humans. However, when volumes of *opinions* are many, human processing becomes a challenge, hence the need for automated processes to extract sentiments from a variety of sources that keep growing in volume, complexity and diversity.

2.2 Sentiment Analysis Basics

Sentiment Analysis can be performed at many levels and at different complexity standards. According to Bing Liu [133], the following are the most common approaches to this topic and the most used techniques in the domain.

2.2.1 Level of Analysis issues

According to Liu [133] there are commonly 3 different levels of analysis:

- Document level.
- Sentence level.
- Entity and Aspect level.

Depending on *how deep one would like to get into a specific technical issue* one would have to decide the level of analysis that will become the focus of the research. Kumar & Sebastian [123] represent graphically the levels of sentiment analysis, using a slightly different approach, as in Fig. 2.1 below.

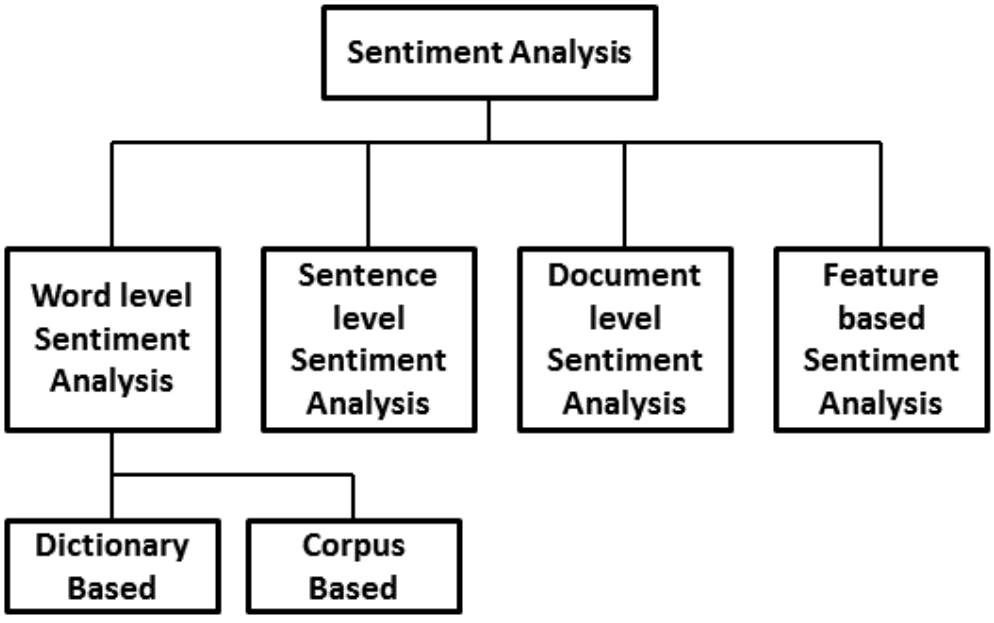


Fig. 2.1. Granularity Levels of Sentiment Analysis

Let us detail a bit more what each of those boxes in Fig. 2.1 mean in terms of Sentiment Analysis. The definitions that follow have been mostly taken from [123, 129].

- Document Level: in this case we consider the document under analysis as a basic unit for which we need to determine a sentiment orientation. In order to do this, some assumptions are made, like presuming that

every text in the document has got as object of its opinion the same object, and that the opinion-holder is the same as well. Clearly, there could be problems with these assumptions. For example: what if the document is a mix of opinions with several opinion-holders? We will **not** develop this case at the document level in this report, as our research focuses on SA at the sentence level.

- Sentence level: as addressed by [123], at sentence level, research has been done on detection of subjective sentences in a document and then, the sentiment orientation of these subjective sentences is determined. It seems that the main challenge at this level of analysis is that both, positive-meaning and negative-meaning sentences, may contain opinion words, making the differentiation process really hard. We *have focused* our research on SA at the sentence level and it will become a central aspect of this report.

- Word Level: in this situation, we usually focus on looking for the 'adjective' part-of-speech. Nevertheless, 'adverbs, nouns and verbs,' could as well convey a sense of subjectivity and could carry opinions. Kumar & Sebastian [123] mention that the automatic annotation of sentiment at the word level are either dictionary-based approaches or corpus-based approaches. We *are covering* this option in our research as it is a building-block for SA at the sentence level. See Section 2.2.2 for more details.

- Feature based: the common example to illustrate this level of sentiment analysis is to consider a *product review* where the author *talks about the positives and negatives of a product. As expected, the reviewer may like some features while dislike others, however the general opinion of the product may be positive or negative* [123]. The other levels of sentiment analysis described above cannot address the complexities of the example mentioned. Hence, the opinion analysis process must be conducted at a feature based level. We are **not** covering this option in our research.

2.2.2 Sentiment Lexicon & its challenges

A sentiment (opinion) lexicon is defined as a list of positive and negative opinion words or sentiment words for a specific language (Our case: English) [102]. It is assumed that such a lexicon could be built as well for any other language that one desires to use. According to [82], **the sentiment lexicon** *is the most important resource for most crucial analysis algorithms*.
Weichselbraun et al. [230] addressed the importance of context when producing sentiment lexicons due to their perception that automated systems have a marginal ability to resolve ambiguities. Wilson et al. [237], Lau et al. [126] and Lu et al. [81] are quoted in [230] and they provide respectively ample support for the importance of context on the potential polarity of words. They argue that language models based on inference tend to do better than other models that do not count with the ability to process context.
As the reader has already probably guessed, the importance of producing an accurate sentiment lexicon is that any polarity / sentiment evaluation to be performed will be based on such a lexicon. In Chapter 15, Section 14.1.1 we will explain how to enrich a sentiment lexicon with words with good-quality polarity scores and part-of-speech (POS) tags.

2.2.3 Natural Language Processing (NLP) issues

We must always keep in mind that *sentiment analysis* is a Natural Language Processing (NLP) problem. As such, many of the issues in NLP are as well problems that must be addressed when dealing with sentiment analysis problems. In [26], Bird et al. address the need for a NLP toolkit that could be used efficiently in education and research, providing as well the basic information required to start doing NLP, and the NLP features

included in the so-called *Natural Language Toolkit*, or *NLTK* (NLTK is a platform for building computer programs in Python -a multi-paradigm high level programming language, designed by Guido van Rossum, in the late 1980s- to work with human language data). According to Bird et al. [26] "NLTK is suitable for linguists, engineers, students, educators, researchers, and industry users alike". Some of the sub-problems that still are the object of further research attention by the NLP community are:

1. Coreference resolution, as mentioned in [133] (not part of our research).
2. Negation handling, as mentioned in [133] (addressed in our research in Section 14.1.2.1).
3. Word sense disambiguation, as mentioned in [133] (addressed by our research in Chapter 14).
4. Meaning extraction (partially addressed in our research when sentiment polarity is extracted; see Section 14.1.2).
5. Optimised parsing, using both, statistical/stochastic and fuzzy approaches (not part of our research).

As we can see, the proper resolution or any added improvements to any of the challenges above described will have a positive effect in advancing the understanding of the sentiment analysis problem. If we consider the stages of analysis in processing natural language as depicted by Robert Dale in his article published in the Handbook of Natural Language Processing [67], see Fig 2.2, there is a lot of room to better certain stages of the process as shown in the graphic just mentioned, as at least three -if not four- of the stages presented in the graphic below, from bottom to top, would be very influential in any sentiment analysis process that could be incorporated once the syntactic analysis stage has been at least partially completed.
Let us expand a bit on the contents of the boxes shown in Fig. 2.2.

- Tokenization: converts a string of characters into words, symbols, sentences or other items conveying some sort of meaning, called *tokens*.
- Lexical analysis: usually deals with generating a *lexicon* and with applying *tagging* to the tokens already generated in the previous step. Most often, the tagging process is called *Parts of Speech* tagging, or **PoS** (parts of speech are nouns, pronouns, adjectives, conjunctions, verbs, adjectives and other related categories).
- Syntactic analysis: provides a structure for every single sentence in a given text, including *parsing*.
- Semantic analysis: aims to find the meaning of sentences depending on the context.
- Pragmatic Analysis: covers the study of what is intended by a speaker and how it could be interpreted by the listener.

If it is true that most of the *open issues* with NLP belong in Semantic & Pragmatic analysis, there is room for improvement as well in the Lexical & Syntactic analysis phases, with *Tokenization* being somehow in a mature state at this point. *Lexical & Syntactic* analysis, through PoS, Parsing and Lexicon generation are key topics if one desires to be successful in Sentiment Analysis. Hence, we will consider the latter tasks as elements to be further analysed (see Appendix D: Data Preparation & Processing and Section 14.1.1).

2.2.4 Media selection issues

The media upon which Sentiment Analysis is applicable is extensive and it has come of age because of Web 2.0 applications. As such, possible media or sources for Sentiment Analysis are, among others:

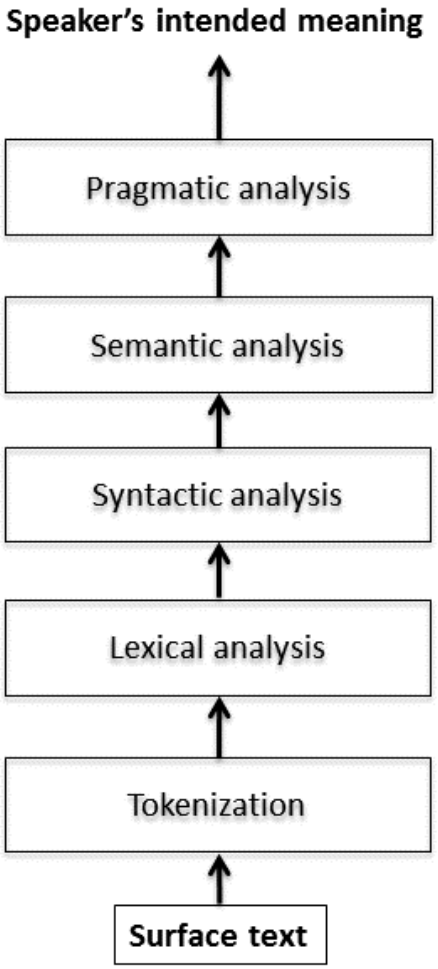


Fig. 2.2. The stages of analysis in processing natural language

- Review sites (Products Reviews by costumers, etc.).
- Web-logs (better known as Blogs).
- Forums.
- Social networks (Twitter, LinkedIn, Facebook, etc.).
- Newspapers articles.
- Stock market & Financial (national and private financial institutions alike) Reports.
- Polls, Political Communiqués and/or Political Reports or Analysis.
- E-mails and internal communication in corporations.
- Others.

2.2.5 Opinions: formal definition

Opinions are easy to understand for human beings, but it is not that easy for a computer to have the same level of understanding. As such, we must try to define formally 'an opinion' so we know what we are talking about. Bing Liu [132] defines an opinion. In an opinion we find the following items:

1. Opinion targets: entities and their features.
2. Sentiments: positive or negative.
3. Opinion holders: persons who hold the opinions.
4. Time: when opinions are expressed.

Opinions then can be: *Regular*, either (a) Direct opinions, (b) Indirect opinions, or *Comparative Opinions*. A regular opinion -which is the type of opinion we are concentrating on for our research, is defined as a quintuple:

$$(e_j, a_{jk}, so_{ijkl}, h_i, t_l) \tag{2.1}$$

where

- e_j is a target entity.
- a_{jk} is an aspect/feature of the entity e_j.
- so_{ijkl} is the sentiment value of the opinion from the opinion holder h_i on feature a_{jk} of entity e_j at time t_l. so_{ijkl} is *positive*, *negative* or *neutral*, or more granular ratings.
- h_i is an opinion holder.
- t_l is the time when the opinion is expressed.

Bing Liu [132] provides a numbers of caveats to this definition, though:

- Although introduced using a product review, the definition is generic enough, in the sense that is applicable to other domains, e.g. politics, social events, services, topics, etc.
- (e_j, a_{jk}) is also called the *opinion target*
- The five components in $(e_j, a_{jk}, so_{ijkl}, h_i, t_l)$ must correspond to one another.

In our research, we will focus on obtaining the third component (so_{ijkl}) in Liu's definition of an opinion 2.1, using only the first two components (e_j and a_{jk}), regardless of whom the opinion holder is and the time at which the action was performed.
Taboada et al. [208], establishes an important differentiation in clarifying the concepts of semantic orientation and sentiment analysis. In [208] the authors refer to *sentiment analysis* as "the general method to extract subjectivity and polarity from text", whilst *semantic orientation* is defined as "the polarity and strength of words, phrases, or texts". These concepts and Liu's formal definition of Sentiment Analysis are key to the understanding on this discipline. Bing Liu [132] describes the Sentiment Analysis task as requiring to "Structure the unstructured", as Natural Language is regarded as unstructured data. According to Liu, the problem definition should provide *a structure* to the unstructured problem.

- Key tasks: identify key tasks and their interrelationships.
- Common framework: provide a common framework to unify different research directions.
- Understanding: help us understand the problem better.

Again, according to Bing Liu [132] in general terms the problem of Sentiment Analysis has two different abstraction aspects: (1) Opinion definition (which we have already addressed in 2.1, and (2) Opinion summarization (opinions are subjective, and we need opinions of a significant amount of people, hence, some kind of summarisation will be required).

2.2.6 Key tasks to be performed in SA

To summarise, the main components of the process of extracting *sentiment* from a given source, as taken from Kumar & Sebastian [123], are:

- Subjectivity Classification: As per [123], a document is a collection of sentences that may, or may not, express the author(s) opinion -those are called *subjective*-. The sentences that are factual in nature are called *objective*. Usually, both types are present in a document. Subjectivity classification is [123] "the task of classifying sentences as opinionated or not opinionated". As such, $S_S \cup S_O = S$, where S represents all sentence in a given document, S_S is the set of subjective sentences in S, and S_O the set of objective sentences in S.

- Sentiment Classification: after finishing the task of identifying whether a text is *opinionated*, then the **polarity** of the opinion must be found. Usually, classifying an opinion as either positive or negative is enough (*Values = [positive, negative]*). However, sometime a multi-class classification might be used, with possible values of *Values = [extremely negative, negative, neutral, positive, extremely positive]*. This is the focus of our research.

- Complementary Tasks.

  - Opinion Holder Extraction: depending on the type of application of sentiment analysis, it would be necessary to identify the *opinion holder*. In some types of documents, there could be multiple opinion holders expressing their opinions about different subjects, hence the need to identify in those cases who is the opinion holder in every case.
  - Object/Feature Extraction: a task that may be necessary to execute -or not- depending of the type of document being processed, is the identification of the *target entity* about which opinions are being issued. For instance, in social media is not uncommon that a number of issues (i.e. blogs) may be addressed, so it is key to get to know about which object/feature opinions are being expressed.

A graphical representation of the described SA tasks follows in Figure 2.3 (dashed boxes represent optional tasks). Even though Kumar and Sebastian [123] do not discuss specifically the concept of graduality of po-

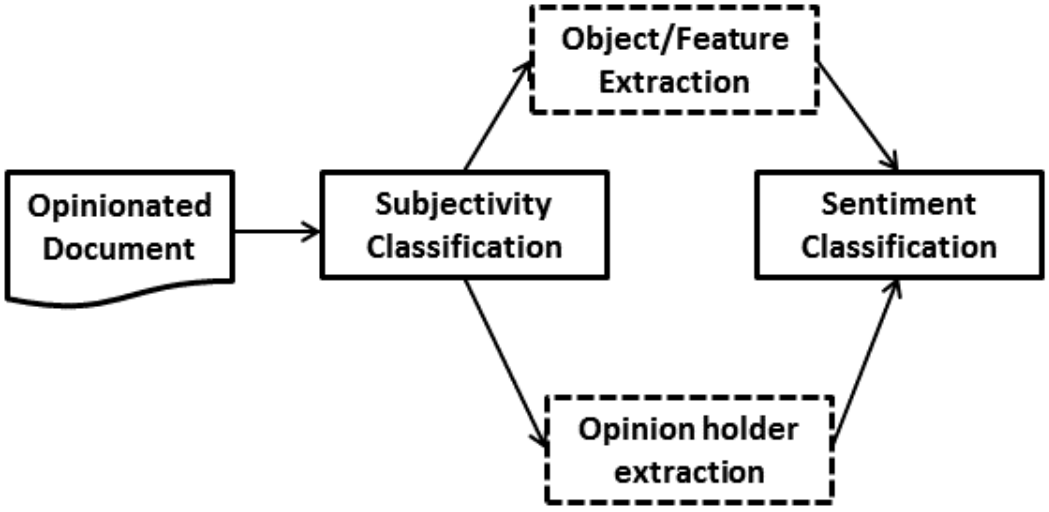


Fig. 2.3. Tasks of Sentiment Analysis

larity, we believe that computing the intensity of the subjectivity polarity would be valuable and should be an important component of the fundamental tasks of sentiment analysis.

2.3 NLP, Machine Learning (ML) and SA

Machine Learning is a discipline that is extensively applied in the field of Sentiment Analysis. Bing Liu, in [133], mentions that "We probably relied too much on Machine Learning" when referring to how limited our understanding is about the Sentiment Analysis problem. Having said that, Machine Learning has played a fundamental role in both, NLP and Sentiment Analysis (SA). In [129] Bing Liu mentions the different *machine learning* approaches applied in Sentiment Analysis while Kumar & Sebastian address multiple issues in Sentiment Analysis and covered Machine Learning techniques in [123, 124] respectively. In [165] Pang et al. address specifically the use of Supervised Learning techniques and Turney [212] covers an example of an Unsupervised Learning approach. All in all, Machine Learning (ML) has become a fundamental tool in Sentiment Analysis, as it grew out of the text mining and classification discipline. As such, we will cover some fundamental aspects of ML in this document in Section 8.1. Kumar & Sebastian [123] quote "Most researchers have defined the Sentiment Analysis problem as essentially a text classification problem and machine learning techniques have proved their dexterity in resolving the sentiment analysis tasks". And then they continue by saying that machine learning techniques require the representation of the key features of text or a documents before doing the processing. These key features are represented as *feature vectors* which are used for the classification algorithm. According to [123], the main difference between *supervised* and *unsupervised* approaches are:

- *Supervised Learning* (Chapter 9): "Machine Learning classification relies on the training set used, the available literature reports detail classifiers with high accuracy, but they are often tested on only one kind of sentiment source, mostly movie review, thus limiting the performance indication in more general cases".
- *Unsupervised Learning* (Chapter 10): "use sentiment driven pattern to obtain labels for words and phrases".

Considering the importance that most researchers assign to Machine Learning, we have further investigated the topic as reported in Chapters 8, 9 and 10.

2.4 Other research directions for SA

There are some other directions in the research world that could be explored with the intention of applying them to the Sentiment Analysis problem. We will discuss two of them: fuzzy sets / logic (which has been used previously in NLP) and Sentic Computing (which is a newer approach; see Chapter 14).

2.4.1 Fuzzy Sets / Logic contribution to NLP and SA

In brief, there have been some successful applications of Fuzzy Sets/Logic theory to both, NLP and Sentiment Analysis. In the literature we find research in the use of Fuzzy Logic in *Anaphora Resolution* (given an expression S_i, its interpretation depends upon another expression S_j in context). As a reference, see the work of Witte & Bergler [241]. Subasic & Huettner [205] used fuzzy sets to analyse affect in text by introducing the concept of *fuzzy semantic typing*. Named Entity recognition has been addressed as well using fuzzy techniques,

as evidenced by the research of Kanagavalli & Raja [112]. At the same time, Nadali et al. [155] address the problem of sentiment classification of customer reviews using a fuzzy sets approach. As fuzzy techniques are at the heart of our research, we will address the topic in full details in Chapter 12.

2.5 Multiple research paths in SA still not explored in full

It is clear that the challenges present in the Sentiment Analysis discipline are many. According to some key researchers in the area (Liu [130, 133], Feldman [82], Pang & Lee [164] and Manning et al. [139], among others) the future of the research in this area lies on exploring as many options as possible among the many challenges available and explore many sub-domains (customer reviews, politicians blogs, marketing sites, company's opinion boards, etc.).
According to Cambria et al. [43] "Mining opinions and sentiments from natural language is challenging, because it requires a deep understanding of the explicit and implicit, regular and irregular, and syntactical and semantic language rules. Sentiment analysis researchers struggle with NLP's unresolved problems: co-reference & anaphora resolution, negation handling, named-entity recognition, and word-sense disambiguation. Opinion mining is a very restricted NLP problem, because the system only needs to understand the positive or negative sentiments of each sentence and the target entities or topics. Therefore, sentiment analysis is an opportunity for NLP researchers to make tangible progress on all fronts of NLP, and potentially have a huge practical impact."
There are many topics of interest and challenges in Sentiment Analysis. Ronen Feldman [82], in his April 2013 article, focused on what he called the five specific problems in the field of sentiment analysis; namely:

- Document-level sentiment analysis.
- Sentence-level sentiment analysis.
- Aspect-based sentiment analysis.
- Comparative sentiment analysis.
- Sentiment lexicon acquisition.

According to Feldman [82], the architecture of a generic sentiment analysis system would somehow look like the graphic presented in Fig. 2.4. At the same time Bing Liu [130] claims that the main technical challenges for the multi-faceted problem that Sentiment Analysis represents, can be found among the topics below, which are described with the help of the following paragraphs of text as an example:

"(1) *Yesterday, I bought a Nokia phone and my girlfriend bought a moto.* (2) *We called each other when we got home.* (3) *The voice on my phone was not clear.* (4) *The camera was good.* (5) *My girlfriend said that the sound on her phone was clear.* (6) *I wanted a phone with good voice quality.* (7) *So I was satisfied and returned the phone to BestBuy yesterday*"

1. *Object identification*: discovering what is the object about which an opinion has been provided. In the paragraph used as an example the objects are *Motorola*, abbreviated as 'moto' and *Nokia*. The noun 'BestBuy' corresponds to the name of the store; hence, it is *neither* part of the comparison processed that the reviewer is providing nor an object in terms of the products' comparison.
2. *Feature extraction and synonym grouping*: the features comment on in our example are 'voice', 'sound' and 'camera'. According to [130] "Although there were attempts to solve this problem, it remains to be

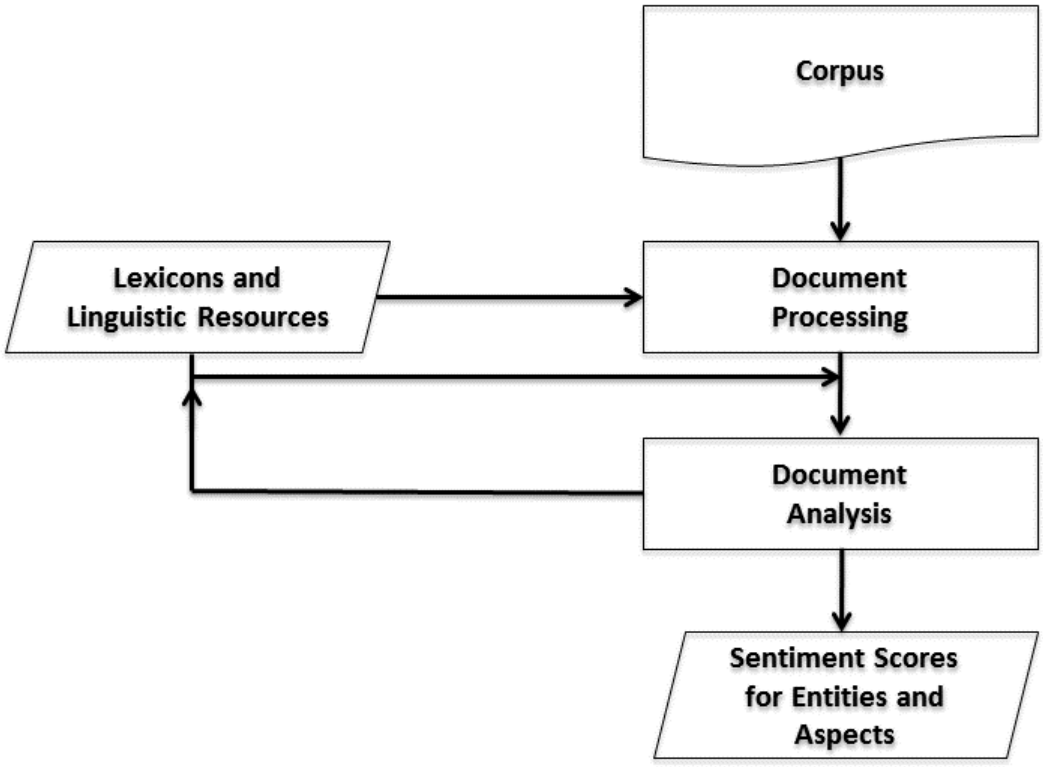


Fig. 2.4. Architecture of a generic sentiment analysis system, as per Feldman

a major challenge". In addition, a feature can be referred to in different ways, i.e. 'voice' and 'sound' refer to the same feature in our example above.

3. *Opinion orientation classification*: the objective of this task is to find out whether there is an opinion on a feature in a given sentence. If there is one, is it positive, negative or neutral? Again, Bing Liu [130] claims that existing approaches are more often than not based on supervised and unsupervised methods. One of the key issues is the identification of opinion words and phrases, as they are key in sentiment analysis. It seems that the main challenge is that there are an unlimited number of expressions that people could use in order to express opinions, and depending on the domains they can be very different. Even when dealing with sentences in the same domain, same words may convey different opinions in different contexts.

4. *Integration*: "Integrating the about tasks is also complex because we need to match the five pieces of information in the quintuple. That is, the opinion oo_{ijkl} must be given by opinion holder h_i on feature f_{jk} of object o_j at time t_l", as per [129, 130]. The quote just presented corresponds to the definition of a *direct opinion* as "a quintuple $(o_j, f_{jk}, oo_{ijkl}, h_i, t_l)$, where o_j is an object, f_{jk} is a feature of the object o_j, oo_{ijkl} is the orientation of the opinion on feature f_{jk} of object o_j, h_i is the opinion holder and t_l is the time when the opinion is expressed by h_i". In addition, Bing Liu mentions that the fundamental problem here is that NLP techniques that still need improvement must be applied to resolve challenges like parsing, word-sense disambiguation, and co-reference resolution. The example provided in [130] are: (i) understanding -on the example paragraph above- depending on the context what is 'my phone' and 'her phone' in sentences (3) and (5); (ii) to which phone does the camera belong to?; (iii) in sentence (4), "The camera was good", we do not have a pronoun and neither the sentence mentions a specific phone. According to Bing Liu [129, 130] these are classical examples of *co-reference resolution*, the

latter being a problem that, despite the fact that it has been studied by the NLP community for a long time, still does not offer an accurate resolution.

2.6 Chapter Summary

In closing this chapter we would like to share the sub-topics that according to those that seem to be the most recognised researchers in the area [43, 82, 102, 129, 130, 133, 164], are still key challenges for the sentiment analysis discipline:

- Named Entity Recognition: what is the person actually talking about. A common example used in the literature is the title of the movie "300". i.e. It is referring to a group of Greeks or a movie?
- Anaphora Resolution: expressed in a more direct way, what a given pronoun or a noun phrase refers to. Let us look at the sentence "We watched a film and went to dinner; **it** was awful". What does "it" refer to?
- Parsing: what is the subject and object of the sentence? Which one does the verb refer to? Which one does the adjective actually refer to?
- Sarcasm & irony identification: Let us review the sentence "Great phone! The battery last a couple of hours". Clearly, the battery life is bad, and the reviewer is offering a negative recommendation about the phone being mentioned.
- Use of abbreviations, poor spelling, punctuation or grammar, lack of capitalisation, etc.
- Sentiment (Opinion) Lexicon acquisition.
- Sentiment Polarity and its *graduality* or intensity.
- Negation handling.
- Aspect-based & Comparative sentiment analysis.
- Effective Classification of multiple opinions.

In this chapter we have covered the main ideas behind Sentiment Analysis, its main characteristics and perceived research challenges. In the next chapter we will address the state of the art in the SA discipline.

Chapter 3

State of the Art in Sentiment Analysis

> "Salomon saith, There is no new thing upon the earth. So that as Plato had an imagination, that all knowledge was but remembrance; so Salomon giveth his sentence, that all novelty is but oblivion."
>
> Francis Bacon (1561-1626). Essays, Civil and Moral. The Harvard Classics. 1909-1914. LVIII-Of Vicissitude of Things.

A significant part of the content of this chapter was used in an article published by the author in 2015 (see reference [10]). As Opinion Mining sits at the confluence of several sub-disciplines -see Chapter 2- (fundamentally NLP, Computational Linguistics, Text Data Mining and AI) its origins cannot be tracked down to a specific date, but rather to a collection of moments in time that defines progress in the sub-areas mentioned above. Most of the important work in syntax and formal languages is attributed to Noam Chomsky [57, 58] and his revolutionary work that occurred between the late 1950s and the late 1960s. Chomsky laid down the bases for modern languages & grammar theory, syntax theory and the concept of transformational grammar as well. In turn, these advances led to improvement in the automatic processing of syntax and grammars by using productions and recursive calls. Parsing and Compiling theory, that today is taken for granted by many, was positively influenced by the work of Chomsky and others that followed.

Already in 1872, Charles Darwin had published his work 'The Expression of the Emotions in Man and Animals', where he mainly addressed aspects of behaviours that are genetically determined. This is probably the first work related to determining the origin and characteristics of emotions. Many other authors, mostly in the Psychology camp, have augmented since then the knowledge we have today about emotions as a fundamental human trait.

In 2013, the authors of [9] attempted to provide some insights in the evolution of Opinion Mining / Sentiment Analysis. In principle, they see the evolution of Opinion Mining to have happened in phases, as follows:

1. Text interpretation phase.
2. Low-level opinion annotation phase.
3. Difference between subjectivity and objectivity phase.
4. Web data mining applications phase.
5. Lexical resources phase.

In the following sub-sections we will focus on showing how the discipline has evolved, but at a higher level than the work of Anbananthen & Elyasir [9]. We would like to approach this summary both ways, chronologically and by the techniques that have been introduced into the discipline of Opinion Mining as we understand it today. As we approach the progress in this discipline by decades, it will become apparent whether the techniques incorporated are related to text interpretation, lexical resources, or otherwise.

3.1 Research time-line in SA/OM

3.1.1 1970 through 1979

The 1970s witnessed a lot of progress related to refining syntactic techniques and generating more advanced parsing and compiling ideas that did reflect in more efficient algorithms. Making sure the proper parsing tree is generated is a fundamental step before more complex tasks can be started. In this arena, the work of Hopcroft [101] and Aho & Ullman [5, 6] is decisive, despite the fact that it concentrates in programming languages instead of natural languages. It is reasonable to say that Aho, Hopcroft & Ullman brought rigour and formality to the parsing techniques world.

3.1.2 1980 through 1989

It is possible to argue that no remarkable work applicable to Opinion Mining was done until the 1980s. The work of Banfield [23] seems to have been instrumental. The so-called Banfield's theory "proposed the use of subjective and objective sentences as indicators, searching the text by providing simple queries and using the psychological element as an important factor for natural language" as mentioned by [9]. In 1983, Winograd [240] published work on language as a cognitive process that started a wave of further research into the cognitive aspects of emotions. In 1987, Ortony et al. [160] published a landmark article called *The psychological foundations of the affective lexicon*, along with his book *The Cognitive Structure of Emotions* [161] published in 1988. These pieces of work are a common reference to building an affective/sentiment lexicon (see Section 14.1.1) and have become important pieces of the puzzle in Opinion Mining.
Let us list the main concepts and/or techniques introduced during this decade in the context of computer assisted activities in NLP somehow leaning towards what today we call Opinion Mining.

- *Word sense*: a word may have multiple meanings and the context would define the right one [236].
- *Subjectivity Vs. Objectivity*: categorisation of sentences into subjective or objective -or even neutral-. Original concept apparently firstly documented by Ann Banfield [23].
- *Strategies for Natural Language Processing*: Carbonell [47] produced an important piece of work related to using computational techniques in NLP.
- *Structure of Emotions*: serious work concerning the structure of emotions was performed and findings published. Morgan & Heise [153] introduced a number of important concepts that would have key ramifications in this research field.
- *Cognition and emotions*: Ortony et al. [161] produced the seminal work entitled "The Cognitive Structure of Emotions" that defines a new path from the psychological perspective on emotions and the cognitive process associated to them.

3.1.3 1990 through 1999

The work of Charniak [50, 51] and separately the contributions of Abney [1] and Manning & Schütze [140] were very influential as they incorporated the use of statistical methods in Natural Language Parsing, Linguistics and more generically into NLP. In [95], Hearst et al. present the so-called 'Text-Based Intelligent System' that focuses on the concept of directionality (e.g. is the agent in favor of, neutral, or opposed to a given event?). The authors claim that, with their method, sentence meaning is mapped to a metaphorical model that is self-contained as no external references are required to find the directionality of a given sentence or paragraph. In [232, 233] the concept of extraction of subjective words is articulated properly, to the point that the authors even proposed a method to determine the beliefs of the characters in the narrative, once the subjective terms have been identified. In 1997, Cardie [48] produces a survey of diverse empirical methods used for information extraction. He discussed the architecture of an Information Extraction System and presented concepts -originated by other authors- like tokenization and tagging, sentence analysis, extraction, etc. as well as a good view on the use of a *corpus* or *annotated corpus* and learning algorithms. A landmark paper was published in 1997 by Hatzivassiloglou & McKeown [92] that addresses the possibility of predicting the semantic orientation of adjectives. As it becomes apparent, the semantic orientation of adjectives carry an important weight on determining the semantic orientation of phrases. Towards the end of the 1990s some researchers started looking at the use of fuzzy reasoning in Opinion Mining. Kruse et al. [122] surveyed the application of fuzzy methods in Data Mining. In 1991, Miller and Charles [149] discussed what they called the contextual correlates of semantic similarity, advancing the field even more when they researched the basis of semantic similarities in a given context. In 1990, G.A. Miller et al. [147] gave to the research community WordNet: an on-line lexical database of English. As presented at `http://wordnet.princeton.edu/`, in WordNet [147] "Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The resulting network of meaningfully related words and concepts can be navigated with a browser. WordNet is also freely and publicly available for download. WordNet's structure makes it a useful tool for computational linguistics and natural language processing". This development would prove to be instrumental in the progress of the Opinion Mining discipline. In 1994, Eric Brill [36] published a piece of work about the transformation-based *Part of Speech Tagging*. The concept of Part-of-Speech was pushed forward and new ideas were put on the table in order to improve methods for part-of-speech tagging. This technique would become key to properly identify the different parts and component of sentences in order to build algorithms that would focus on extracting meaning and orientation out of sentences. The main achievements of this decade could be summarised as:

- *Statistical Techniques*: use of statistical techniques in order to improve the quality and accuracy of the process of generating parsing trees and other linguistic tools and artifacts [1, 50, 51, 140].
- *Directionality*: the concept of 'directionality', better known nowadays as *orientation* or *polarity*, is introduced by [95]. Other authors contributed as well, but in principle the origin of the idea can be tracked down to Hearst and the Palo Alto Xerox team.
- *Annotated Corpus*: the introduction of an annotated corpus, tokenization & tagging, as well as other syntactic analysis tools achieve an adequate level of maturity as covered in [48]. Key research work that added to this space are [77, 157], among others.
- *Semantic orientation of adjectives*: the prediction of semantic orientation of adjectives and sentences saw an important development in the late 1990s mainly through the work of Hatzivassiloglou et al. [92].

- *Semantic similarity*: the ideas behind semantic similarities when a given context is provided are advanced [149].
- *WordNet, an on-line lexical database of English*: a lexical database of the English language is created and made public [147].
- *Transformation-based Part of Speech Tagging*: the concepts and techniques associated to identifying part-of-speech efficiently and accurately have matured and their use become more common in different research areas [36].

3.1.4 2000 through 2016

With the beginning of the years 2000s, the Opinion Mining / Sentiment Analysis discipline starts an accelerated development process. Bing Liu is one of the most respected researchers in Opinion Mining. His article, *Sentiment Analysis: A Multifaceted Problem* [130], addresses the complexities and multiple faces that this discipline can show. Then, he published his book *Sentiment Analysis and Opinion Mining* [133] that presents the most updated version of the discipline (2012). Pang & Lee [164] had published four years before their book *Opinion Mining and Sentiment Analysis* that until 2012 was the most complete work in the area. In 2001, Subasic & Huettner [205] released the most important contribution we are aware of to the use of Fuzzy Sets / Logic principles in Opinion Mining. Since then, others have followed in their footsteps, but certainly, Fuzzy Sets Theory has been so far a bit of an outsider in the research field for Sentiment Analysis. Jusoh & Alfawareh [108] released in 2013 research work where they applied fuzzy sets to Opinion Mining and some researchers have started to think about potential applications of fuzzy sets in SA (see Chapter 12). We have mentioned before the importance of WordNet. In 2006, Esuli & Sebastiani [80], published their paper on SentiWordNet, a lexical resource specific to Opinion Mining. SentiWordNet assigns to each synset -sets of synonyms for groups of English words- of WordNet three sentiment scores: positivity, negativity, or objectivity. In 2013, Ronen Feldman [82], attempted to bring Sentiment Analysis to the front-page and published and article in Communications of the ACM, called *Techniques and Applications for Sentiment Analysis* [82]. This article has created a lot of additional attention in the research community. New techniques have been showing up steadily, and Cambria et al. [43], addressed this in their article N*ew* A*venues in* O*pinion* M*ining and* S*entiment* A*nalysis*. In combination with Cambria & Hussain's book [41], the authors proclaim that new techniques, like the so-called *Sentic Computing*, could offer some new lights into the Sentiment Analysis problem (in our research work we will present experimental results based on Sentic Computing techniques [176] in Chapter 14). In 2010, Dzogang et al. [74], published in the *IEEE Fuzzy Systems* journal, their article *Expressions of* G*raduality for* S*entiments* A*nalysis* - A S*urvey*. The team of authors included the presence of the prestigious researcher Bernadette Bouchon-Meunier. In our opinion, the main idea behind the article, was to depict how it is necessary, in order to be successful in Sentiment Analysis and related disciplines, to understand two key factors: (a) the inclusion of the use of some fundamental emotions structure coming from the world of Psychology and (b) the further looking into the potential fitness of fuzzy sets to model *graduality* in a proper way. We will explore the latter further in our research in Section 14.1.3.

Many of the tools utilised in Sentiment Analysis to resolve subjectivity identification and polarity extraction are based on some sort of Machine Learning technique. Most of the literature and bench-marking processes established are based in Supervised Learning. However, some Unsupervised Learning techniques have been very successful as well, as discussed in an influential article by Peter Turney [212]. In this article, an unsupervised technique is described an applied successfully. The content of the paper is about *Semantic Orientation Applied to Unsupervised Classification of Reviews*. The techniques are mostly based on the PMI-IR algorithm

that is used to estimate the semantic orientation of a phrase. As described in Turney's article "PMI-IR uses Pointwise Mutual Information (PMI) and Information Retrieval (IR) to measure the similarity of pairs of words or phrases". One of the main criticisms towards unsupervised methods is the existing difficulty to dynamically enrich the lexicon. Some other methods have been proposed, like the one presented by Banea et al. [22], where a B*ootstrapping* M*ethod for* B*uilding* S*ubjectivity* L*exicons for* L*anguages with* S*carce* R*esources* has been presented. Approximately at the same time, Ding et al. [71] published a paper presenting what they called a H*olistic* L*exicon*. More recently, in 2011, Taboada et al. [208] released an important article in the Computational Linguistic journal, entitled *Lexicon-*B*ased* M*ethods for* S*entiment* A*nalysis*, where they emphasised a number of techniques for generating a quality lexicon. Hatzivassiloglou et al. had published in the 1990s a remarkable article about determining polarity of adjectives. In 2000, they addressed the gradability of subjective sentences based on adjective orientation [93]. Their findings are stressed again by Srivastaval et al. with their article from 2010 [200], *Effects of adjective orientation and gradability on sentence subjectivity*. The latter topic was extensively covered as well by Wilson, Wiebe & Hoffmann in 2005 with their paper *Recognizing Contextual Polarity in Phrase-level Sentiment Analysis* [238]. In this case, beyond using adjectives as the main part-of-speech to attempt recognising subjectivity, they look at the context as well using some specific techniques they proposed in their work. In 2009, Wiebe et al. [239], addressed again the recognition of contextual polarity with their article *Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis*. The focus of the research is at the sentence / phrase level.

When one attempts to establish the orientation of the sentiment in a document, one is faced with the need for summarising somehow all the content. Hu & Liu [102] published their article *Mining and summarizing customer reviews* with the idea of addressing some techniques to effectively summarising opinions. On the same topic, Suanmali et al. [203] proposed a fuzzy logic based method for improving the summarisation of text. Pang et al. [165] released to the research community their paper *Thumbs up?* S*entiment* C*lassification using* M*achine* L*earning* T*echniques*. The emphasis in this article is in supervised machine learning techniques, that as mentioned before, have become the most used tools in Sentiment Analysis until today. On that subject, in 2012, Kumar & Sebastian [124] presented a very well documented survey on Supervised Machine Learning. Again, the central time in the article is devoted to Supervised Learning tools, highlighting the apparent benefits of using Support Vector Machines (SVMs). Realising as well that determining subjectivity is such an important aspect of Opinion Mining, Bing Liu contributed in 2010 with a chapter to the *Handbook of Natural Language Processing*. His article was entitled *Sentiment* A*nalysis and* S*ubjectivity*, stressing one more time the importance of differentiating between objective and subjective sentences. If it is true that it is less common, some researchers have looked into the possibility of applying semi-supervised methods in machine learning, like the case of Dalal & Zaveri [65], where they introduce the use of semi-supervised learning for opinion summarization and classification for online product reviews.

In 2013 we see a cluster of interest in applying *fuzzy techniques* to our research topic. Either to assist in resolving ambiguity in text, as the work of Kanagavalli & Raja [112], or the article published by Ahmad & Rana [4] that offers a review of *Fuzzy sets in* D*ata mining*. The latter, is more focused in the data mining discipline, but it offers some insights on how fuzzy sets are used in a domain that is so tightly related to Sentiment Analysis. A more focused approach in Sentiment Analysis using fuzzy sets, is the paper by Fu & Wang [88], where fuzzy sets are used to model sentiment classification at the sentence-level for Chinese. In addition, in 2007, Khoury et al. [116] released their article on using fuzzy set theory to drive the semantic understanding of general linguistic items. Dalal & Zaveri [66] introduce in 2014 a method based on *fuzzy linguistic hedges* for opinion mining in online user product reviews. At the moment of writing this paper, this one seems to be most recent contribution of fuzzy methods to Opinion Mining / Sentiment Analysis.

Later on, some mixed-methods start to flourish. For example, we have the work of Liu & Tsou [134], *combining a large sentiment lexicon and machine learning for subjectivity classification*. In this article the focus is in using Supervised Machine Learning methods jointly with a qualified sentiment lexicon. We believe that the future of research in Opinion Mining is probably bound to lean towards hybrid methods. On the same token, Wiebe & Riloff [235] proposed a method to do simultaneously subjectivity analysis and information extraction. Their claim is that doing one enables the other. In 2011, the interest in fuzzy methods flares up again, with the publishing of a research by van der Heide et al. [219], *Computational Models of Affect and Fuzzy Logic*. As the title suggests, the authors cover the topic of modelling *affect* through applying fuzzy logic. Other researchers have focused on using fuzzy logic only for determining the strength of a pre-established opinions in web reviews. That is the case of the work from 2011 by Kar & Mandalof [113]. An attempt to combining together two or more methods is due to Poria et al. [176].

For some time, the focus of identifying subjectivity by analysing adjectives was the standard. However, verbs and nouns are capable as well of conveying an emotion or sentiment. The work of Maks & Vossen [136] focuses on exploring the analysis of verbs to create a *verb lexicon* that would aid on establishing sentiments in opinion mining applications. Subrahmanian & Reforgiato Recupero [206] present a paper where they combine adjective, verbs and adverbs in a effort to improve subjectivity classification. The article is called AVA*:* A*djective-*V*erb-*A*dverb* C*ombinations for* S*entiment* A*nalysis*. We believe that using all of these three components of part-of-speech simultaneously may contribute to a more reliable subjectivity polarity classification process. In 2012, Nguyen et al. [249] divulge their paper *Linguistic Features for Subjectivity Classification*. The authors reason as follows: "features play the most important role for getting accurate subjective sentences. In this paper, we will enrich features by using syntactic information of the text. From our observation when investigating opinion evidences in the texts, we will propose syntax-based patterns which are used for extracting rich linguistic features" [249]. As they combine the new features with conventional ones obtained from already established research lines of work, their method can be considered somehow as a *hybrid* approach. Mudinas et al. [154], presented their article *Combining Lexicon and Learning Based Approaches for Concept-level Sentiment Analysis* in 2012. They claim that their proposed system combines together concept-level sentiment analysis and opinion mining lexicon-based and learning-based approaches. This is another sample of researchers attempting to come up with hybrid approaches. Then, Lotfi Zadeh [266, 268] put forward the concept of Precisiated Natural Language (PNL). Zadeh argues that "A Natural Language is basically a system for describing perceptions". What does Zadeh mean by precisiated? As per [268], "precisiated in the sense of making it possible to treat propositions drawn from a natural language as objects of computation?". Despite the fact that the idea is very tempting, not too much additional research has been conducted in PNL -as far as Sentiment Analysis goes-. We have decided to include it in this summary, as it puts forwards a concept that could possibly blossom in the Opinion Mining arena. Wei Wei [229] released in 2011 an article entitled *Analyzing Text Data for Opinion Mining* that addresses the space of entity-related opinion detection and sentiment ontology trees. This is a departure of most of the methods we have presented above, but we wanted to show it as another possibility that some researchers are considering.

Dealing with metaphoric language is hard, and some researchers have spent some time suggesting how to address the problem. Recognising irony and sarcasm are tough topics as well, and some work has been done in this area. These topics are very important when dealing with opinions, particularly when the text being analysed has got some politics content. Vanin et al. [220] released an article in 2013 providing some clues into irony detection in Tweets. Shutova [195] proposed in 2010 models for understanding metaphors in NLP, whilst in 2013 Bollegala & Shutova [30] investigated the option of interpreting metaphors using paraphrases extracted from the web. Rentoumi et al. [182] have investigated more specifically, metaphorical language in sentiment

analysis (2012). In 2015, Hogenboom et al. [100] focus in using rhetorical structure in sentiment analysis, and utilises structural aspects of text as an aid to distinguish important segments from those less important, as far as contributing to the overall sentiment being communicated. As such, they put forward a hypothesis based on segments' rhetorical roles while accounting for the full hierarchical rhetorical structure in which these roles are defined. Heerschop et al. [96] propose a Rhetorical Structure Theory (RST) based approach [138], called *Pathos*, to perform document sentiment analysis partly based on the discourse structure of a document. Text is then classified into important and less important spans, and by weighting the sentiment conveyed by distinct text spans in accordance with their importance, the authors claim that they can improve the performance of a sentiment classifier. The idea of applying discourse analysis to determine the parts of the text that are most relevant to the overall document sentiment is obviously relevant and could be helpful in achieving our overall aim by extending the model proposed in this paper. Other articles worth mentioning explore topics around sentiment lexicon-based techniques, like the 2014 contributions of Cho et al. [56] and Huang et al. [103]. The work by Bravo-Márquez et al. [35] (2014) on the use of multiple techniques and tools in SA, offers a complete study on how several resources that "are focused on different sentiment scopes" can complement each other. The authors focus the discussion on methods and lexical resources that aid in extracting sentiment indicators from natural languages in general. A comprehensive piece of work on *semantic analysis* is produced by Cambria et al. [44], while in 2016, Schouten and Frasincar work [194] provides a complete survey, specific to aspect-level sentiment analysis. As already mentioned, a number of researchers have explored the application of hybrid approaches by combining various techniques with the aim of achieving better results than a standard approach based on only one tool. Indeed, this has been done by Poria et al. in [176] where a novel framework for concept-level sentiment analysis, Sentic Pattern, is introduced by combining linguistics, common-sense computing, and machine learning for improving the accuracy of tasks such as polarity detection. The authors claim that "by allowing sentiments to flow from concept to concept based on the dependency relation of the input sentence, authors achieve a better understanding of the contextual role of each concept within the sentence and, hence, obtain a polarity detection engine that outperforms state-of-the-art statistical methods". When no matching sentic pattern is found in SenticNet [45] (2014), they resort to Supervised Machine Learning. Related to the use of lexicons in SA approaches, it is worth mentioning the following two research efforts. The first one is by Hajmohammadi et al. [91] (2015) on a novel learning model based on the combination of uncertainty-based active learning and semi-supervised self-training approaches, which also addressed the challenges associated with the construction of reliable annotated sentiment corpora for a new language. This research provided us with important lessons on the difficulties and potential pitfalls of embracing such a task and how to better deal with it. The other research effort is by Hogenboom et al. [98, 99] (2015) on the use of emoticons as modifiers of the sentiment expressed in text and as vehicles of sentiment by themselves. According to the findings of the authors, the sentiment associated to emoticons dominates the sentiment conveyed by the text fragment in which these emoticons are embedded. In their work they introduce a sentiment lexicon, which is a point of commonality with the research presented here, as well as a cleverly designed emoticon lexicon. In 2016, Appel et al. [13] published an article proposing a hybrid approach to sentiment analysis at the sentence level, which not only establishes sentiment polarity (using semantic rules, improved negation management and an enhanced sentiment lexicon), but also computes the intensity of the sentiment polarity (using fuzzy sets as a fundamental tool). This latter article is based on research conducted and reported in this thesis report document.

3.2 Bibliometrics

We believe that it would be interesting to understand the bibliometric aspects of the work published in the Sentiment Analysis / Opinion Mining area of research. We will attempt to see the portion of work that has been carried out using supervised machine learning methods versus those based on fuzzy set theory. The years of publishings as well as the country where the work has been conducted will be reviewed as well.
Our approach will be founded on a simple search based on keywords (machine learning or fuzzy sets) in the larger context of Sentiment Analysis / Opinion Mining. The tools that will be used are *Scopus* (a large abstract and citation database of peer-reviewed literature by Elsevier B.V.) and the *Web of Knowledge* (an academic citation indexing and search service by Thomson Reuters).
If it is true that the aforementioned search will exclude articles written before the term *Sentiment Analysis* saw the light and became fully accepted -later on in the middle 2000s most likely- it will provide us with good indicators of the numbers of publications on the topic since the middle 2000s as well as the country where the research initiative was conducted and published. Likewise, articles indexed by other sub-topics of Sentiment Analysis, like subjectivity classification or identification, polarity extraction, etc. could have been partially excluded, too. Nevertheless, we believe that based on the review of the literature that we have carried out, the potential exclusion of those articles will not create a deviation in the results we have obtained already. For consistency, we are including *only* articles written in the English language. The results obtained using Scopus and the Web of Knowledge are equivalent, so that we will present below only the results obtained using *Scopus*. The two graphics shown below used the following keywords:

- (Fuzzy Sets) and (Sentiment Analysis): See Figure 3.1
- (Machine Learning) and (Sentiment Analysis): See Figure 3.2

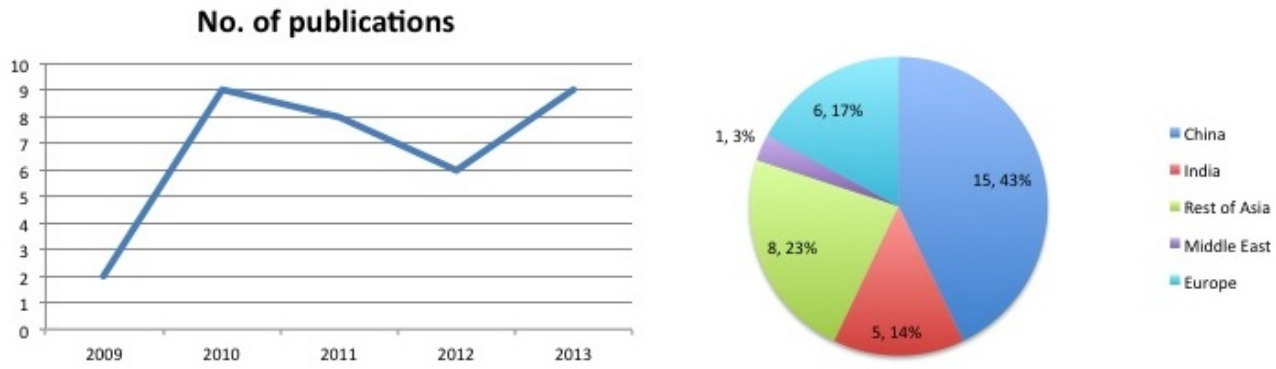


Fig. 3.1. Outcome of search using keywords *Fuzzy Sets* and *Sentiment Analysis*

As we take a closer look to the graphics provided above, we immediately note two characteristics:

1. Research in Sentiment Analysis using Machine Learning techniques depicts a curve that shows primarily a clear trend towards sustained growth, whilst the one representing the utilisation of Fuzzy Sets shows no material growth and a clear lesser number of publications
2. When we look and the countries where the articles have been published we notice that the utilisation of Machine Learning (ML) techniques -more specifically Supervised Machine Learning- is high all across the board, showing the USA, China, India and Europe as clear leaders. If we observe now the utilisation

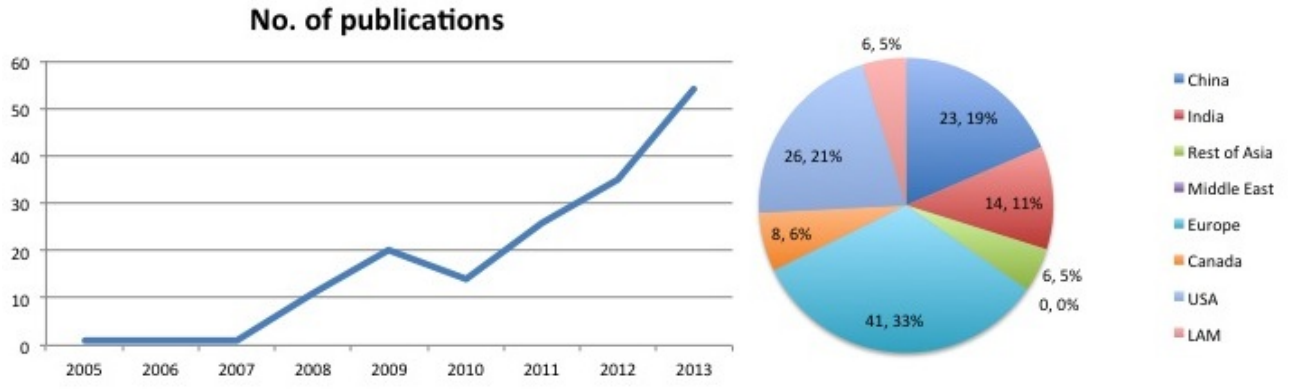


Fig. 3.2. Outcome of search using keywords *Machine Learning* and *Sentiment Analysis*

of Fuzzy Sets to model the Sentiment Analysis problem, the first thing that we realise is that it is mainly an affair mostly pursued in China, India and Europe. In addition, research using ML techniques seems to have started earlier as well

Why is this? One dares to adventure that it is perhaps because a significant number of the researchers with a Computer Science / Mathematics education background involved nowadays with researching SA/OM come from the Text Data Mining and Classification fields where the use of statistical methods have been a well-established tool for some time now. Hence, it would be natural to *export* the same knowledge and techniques and apply them to a new domain that, nevertheless, is somehow related. Or is it perhaps that the utilisation of Fuzzy Sets (FS) techniques have been proven to be not successful? If so, how has *success* been measured? It is interesting to see that in the cases of China and India, and to a certain extent Europe as well, research efforts are present in both camps (ML and FS). However, for the USA, at least for the period of time chosen and the search keywords and data sources utilised, the focus is clearly on the ML camp, despite the fact that one of the most influential papers supporting the use of FS was written in 2001 in the USA [205]. One may think as well that two of the most reputed researches in SA/OM, Bing Liu [133] and Bo Pang [164], have made ML their fundamental tool. For instance, Prof. Liu's early research was in data mining, Web mining and *machine learning*, fields in which he published abundantly (as appears in his Biography in his book [133]).

At this point, we can only draw some conjectures, but we believe that it will not be completely nonsensical to think that the primary research interest of some authors may have migrated to this newer field of SA/OM. Moreover, the use of statistical techniques in NLP and Computational Linguistics are common and have been aptly utilised since the 1990s (see the work of Eugene Charniak [50, 51]).

Fuzzy Sets have been used extensively to model uncertainty and ambiguity, traits that are undoubtedly inherent to Natural Languages and as a consequence part of the challenges inherited by SA/OM. Somehow, Fuzzy Sets may seem alien to the community of Linguistics, with the exception perhaps of the utilisation of Fuzzy Grammars. We conclude then that there are a number of potential reasons that could explain why the use of Supervised Machine Learning techniques has been favoured. However, we have not been able to find so far hard evidence that the utilisation of Fuzzy Sets, perhaps in combination with some other syntactic techniques and even Unsupervised Learning tools, could not yield favourable results. Prof. Bing Liu, a world-wide recognised expert in the area of SA/OM and one the researchers that have attempted to push the limits in the field of Sentiment Analysis, has mentioned that "We probably relied too much on Machine Learning" [129, 130, 133], when referring to how limited our understanding is about the Sentiment Analysis problem in despite of the

recent progress that has been achieved.

As a result of the discussion presented in this section and other arguments to be presented in subsequent parts of this report (Chapters 4 and 14), we do believe there is merit in investigating further the potential use of Fuzzy Sets in the Sentiment Analysis problem; especially in the sub-areas of subjectivity polarity determination and graduality.

Before we close this section, we will share an updated version of Fig. 3.1 and Fig. 3.2, with data covering until the end of 2016. The information extracted from the original graphs contributed to the decision-making process at the time (2014) on which research paths to follow. The updated figures provide an extended view into what has happened in the last couple of years.

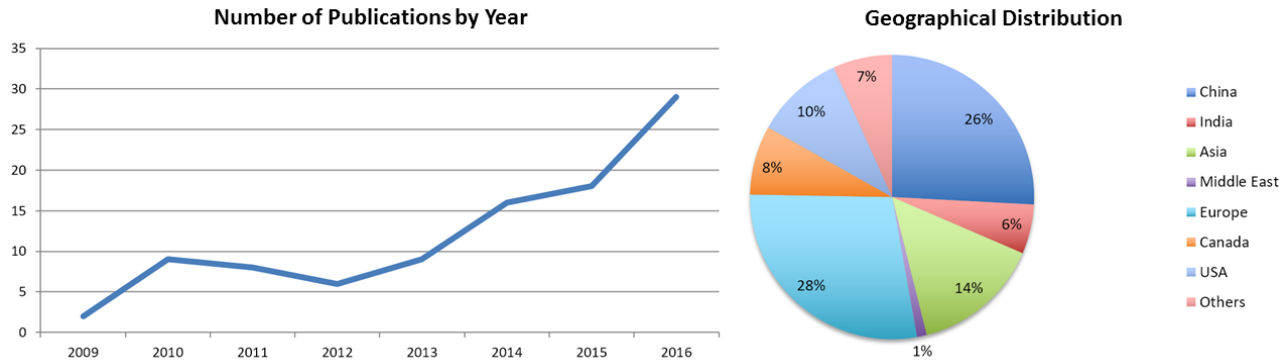


Fig. 3.3. Outcome of search using keywords *Fuzzy Sets* and *Sentiment Analysis* (2016)

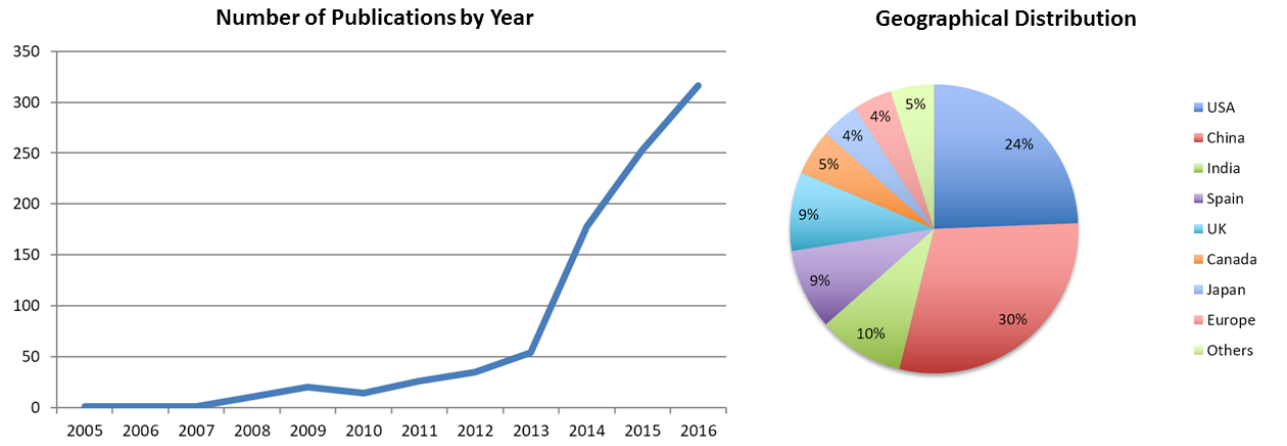


Fig. 3.4. Outcome of search using keywords *Machine Learning* and *Sentiment Analysis* (2016)

As we can appreciate, the proportion of articles published on the SA topic using *machine learning* as a tool of choice continues to be significantly higher that those cases where *fuzzy sets* have been applied instead. Having said that, we observe that the slope of the curve for the latter case has changed showing an upwards trend, indicating more interest in the research community on the possibility of applying fuzzy sets theory in the SA domain.

3.3 Chapter Summary

Let us try to summarise the main aspects covered in this section.

- *A multifaceted problem*: Sentiment Analysis (SA) / Opinion Mining (OM) recognised as a multifaceted problem [130]. The most complete publications on the domain topic published by Bing Liu [133] and Pang & Lee [164], and more recently Ronen Feldman [82].
- *Adjectives, verbs and adverbs combination for subjectivity identification*: in addition to using adjectives as the main component of part-of-speech that influences the subjectivity of a given phrase, verbs, adverbs and nouns are incorporated as well [136, 206].
- *Subjectivity identification improved*: A significant number of researchers have contributed to this aspect. Probably all the references listed in this section. As representative of the progress in this sub-field, we have chosen to highlight the work of Bing Liu and Ronen Feldman [82, 129].
- *Graduality for Sentiment Analysis*: an important topic presented very well by Dzogang et al. [74].
- *Linguistic Lexicon enhanced*: mostly, WordNet & SentiWordNet as covered by [78, 79, 80]. Specific processes for the generation of efficient subjectivity lexicons [22].
- *Utilization of Fuzzy Sets in Opinion Mining*: a number of authors addressed in different fashions the utilisation of fuzzy sets to model OM problems or to contribute to related problems (i.e. opinion summarisation, ambiguity resolution, etc.) [7, 108, 113, 155, 205, 219].
- *New approaches to Opinion Mining*: mainly represented by Sentic Computing and other approaches [41, 43, 46, 229].
- *Metaphor, irony and sarcasm handling*: Sarcasm & irony are hard to detect in an automated fashion. Metaphors are difficult as well as it usually requires context and previous knowledge. Some research has been done in this area, especially in the period 2000-2014. [30, 182, 195, 196, 197, 220].
- *Supervised machine learning techniques*: the most commonly used technique is Supervised Machine Learning. A lot of progress has been achieved. References are abundant. [49, 71, 120, 123, 124, 133, 134, 164, 165, 208, 209, 238, 239] and many others.
- *Unsupervised machine learning techniques*: the most notorious examples of applying Unsupervised Machine Learning techniques have been published by Turney [212, 213, 214]. More recent contributions have been provided by [89].
- *Concept of Contextual Polarity introduced*: polarity influenced by context is supposed to increase the accuracy of the sentiment analysis process. Important work by [238, 239].
- *Precisiated Natural Language (PNL)*: enables the possibility of treating propositions drawn from natural language as objects of computation. The work of Lotfi Zadeh [268].

Combining the present Chapter with Chapter 2 (Part I of this document) we have enough content to establish a solid research framework. We have shared as well some bibliometric data that highlights specific areas of research. As such, we are now ready to explore what has motivated us to pursue the research contained in this report.

A complete *motivation* for our research is presented in Part II, Chapter 4, whilst our research hypothesis is elaborated in Part II, Chapter 5.

Part II

MOTIVATION, HYPOTHESIS & METHODOLOGY

Chapter 4

Motivation

> I think the big mistake in schools is trying to teach children anything, and by using fear as the basic motivation. Fear of getting failing grades, fear of not staying with your class, etc. Interest can produce learning on a scale compared to fear as a nuclear explosion to a firecracker.
>
> Stanley Kubrick

A significant part of the content of this chapter was used in articles published by the author. See references [10, 11, 12, 13, 14, 15, 16, 17, 18]. Let us now discuss the fundamental motivation behind our research in Sentiment Analysis (**SA**) / Opinion Mining (**OM**).

4.1 Aspects that has driven our research

In the article *Expressions of Graduality for Sentiment Analysis - A Survey*, the authors, Dzogang et al. [74], go beyond the most common motivation for Sentiment Analysis and/or Opinion Mining -automate the classification of social media opinions, books reviews, films' rankings, etc.- and attempt to "review methods taking account of *intrinsic psychological models* components of graduality as well as extrinsic components issued from computational intelligence approaches. In particular, beyond psychological models of sentiments that define affective states as multidimensional vectors in affective continuous spaces, we identify three components of graduality, namely composition or blending, intensity and inheritance".
Basically, the authors do a deeper analysis of the origins of emotions and sentiments and investigate among the technical tools at hand, which are closer to the nature of the problem being analysed. A such, they even address the Darwinian vision that "imposes sentiments to be organised as a finite set of of basic affective states, universally shared by all human beings". The latter are usually called *primary sentiments*. In [74], Dzogang et al. make reference to key studies by *R. Plutchik (The Emotions), P. Ekman (An argument for basic emotions) and Cowie & Cornelius (Describing the emotional states that are expressed in speech)*, that discuss emotions from a deep psychological standpoint, and in doing so, they highlight the nature of emotions and their graduality and fuzziness. In fact, in Dzogang et al. discuss briefly that authors in general refer mainly to psychological models when attacking the sentiment analysis problem. However, appraisal event models may be successful at developing "artificial affective agents", as described in [77], [159] and [137]. Dzogang et al. claim that "it must be underlined that some appraisal based approaches make use of graduality through fuzzy inference and fuzzy aggregation for processing affective mechanisms ambiguity and imprecision.". The caveat they make, though, is that these so-called appraisal-based methods are *not* great at *sentiments discrimination*.

As discussed at the *SA Main Concepts* Chapter 2 of this document, there are mainly two recognised main approaches to the problem of extracting sentiment automatically from a given source (see Fig. 8.1). Those are [208]:

- **(a)** the lexicon-based approach, that involves calculating orientation for a document/sentence from the semantic orientation of words or phrases in the source.
- **(b)** the classification approach which involves building classifiers from labelled instances of texts or sentences.

See Chapter 8 for a complete discussion on the lexicon-based and machine learning approaches.
Taboada et al. [208] discuss the most common and accepted approaches to extract sentiment automatically from texts. In Figure 8.1 we have attempted to summarise those, as we will discuss the subject further down the road.
All these arguments make us think that if perhaps not great for deep psychological and physiological analysis, the fact that fuzzy sets can be used successfully to model the ambiguity and imprecision of *affective states*, will make them an acceptable tool for modelling sentiments. If in addition we recall a quote for Bing Liu [133], one of the main world experts in Sentiment Analysis, "We probably relied too much on Machine Learning", we tend to believe that there might be room for pursuing a different avenue than the traditional Supervised Machine Learning approach.

4.2 Chapter Summary

In this chapter we have established the aspects that has driven our research. In the next chapter we will cover the main *hypothesis* that we have formulated.

Chapter 5

Hypothesis

"Not all those who wander are lost."

A line from the poem *All That is Gold Does Not Glitter* by J. R. R. Tolkien, The Fellowship of the Ring.

In this chapter we will establish our main research hypothesis and proposed-solutions as well as the key research questions that will be addressed to achieve our research objective. The material presented here has been partially utilised as well in articles by Appel et al. [10, 13].

5.1 Are there other paths besides Supervised Machine Learning to address the Sentiment Analysis problem?

If we take a closer look to the SA/OM field and think about classification techniques, the first thing that comes to mind is the utilisation of Machine Learning (ML) approaches. Traditionally, in ML we think of unsupervised, semi-supervised or supervised machine learning algorithms. The latter technique, as we well know, relies heavily on training, which implies counting with the adequate and voluminous annotated datasets. This constitutes a drawback and we would like to avoid, if possible, having to count on prior data for training purposes, as this is not always possible, and we would like to explore a path that does *not* require such a massive annotation effort. As a consequence, supervised machine learning does *not* look like a technique that we would be interested in pursuing. Ideally, in the context of SA/OM, an unsupervised strategy would rather "measure how far a word is inclined towards positive and negative" [224].
Ultimately, the problem of SA/OM is basically a NLP problem with emphasis in finding when a sentence reveals an opinion -as opposed to a fact- and extracting the polarity of the opinion (usually, Positive or Negative). Kanaga [112], in discussing ideas presented by Lotfi Zadeh in [269], says "The semantics of natural languages and information analysis is best handled by the epistemic facet of Fuzzy Logic. In the epistemic facet, natural language is viewed as a system for describing perceptions and an important branch of the same is possibility theory and computational theory of perceptions". Hence, does it worth to take a new look to Fuzzy Sets / Logic as a potential effective tool in SA/OM? The path that we would like to pursue, will include the utilisation of linguistic semantic rules, lexicon-based approaches and fuzzy sets as fundamental components of a *hybrid approach towards SA/OM*. Based on the information, references and discussions shown in previous sections, we would like to think that the following concepts could become cornerstone to a potential research direction that would differ from the most commonly followed paths.

1. In Sentiment Analysis the most utilised approach, which accounts as well for most of the research published, is *text classification* (see figure 8.1) relying heavily on Machine Learning techniques, especially

Support Vector Machine (SVM) and Naïve Bayes.

2. Fuzzy Sets and Fuzzy Logic have been used as well, but to a lesser extent, and the literature about it is less abundant when compared to (1) above.

3. One of the main objections with regard to the use of Fuzzy Logic/Sets in Sentiment Analysis is given by Balahur-Dobrescu [21]: "we can show that while the fuzzy models of emotion perform well for a series of cases that fit the described patterns, they remain weak at the time of acquiring, combining and using new information". However, we believe that some of the shortness can be minimised by combining together fuzzy methods and some semantic rules and linguistic techniques. See, for example, the progress reported on acquiring new information on Kruse et al [122] (*using neuro-fuzzy modelling*) and Hüllermeier [104] (*applying learning fuzzy rules*).

4. With the advent of SentiWordNet [79] and Senticnet [42, 45], the availability of solid sentiment lexicons with incorporated updating capabilities has become a reality.

5. Hatzivassiloglou et al [92, 93] proposed a methodology to *predict* the semantic orientation of adjectives that could be extended to nouns, adverbs, and verbs. It seems that predicting the semantic orientation of certain parts of speech can greatly help on suggesting the semantic orientation of sentences and documents.

6. Grammatical dependencies may play a significant role in a proper understanding of a sentence. As quoted from [200], "In any sentence, words are arranged in a proper sequence to communicate information. The complete meaning of a sentence is not only determined by the meaning of words, but also by the pattern in which words are arranged".

7. Supervised machine learning has proven to be a strong classification tool. However, it will depend enormously on the training data and we are attempting to move towards a system that depends less on pre-existing annotated data. We would like to rely more on the richness of fuzzy sets as a modelling apparatus, as well as in semantic rules, syntactic analysis and aggregation techniques (see Chapters 12, 13 and 14).

5.2 Research Questions

The fundamental research question we are posing is whether a **`hybrid approach`** as described in our Hypothesis 1, is well equipped to model subjectivity polarity determination and polarity graduality determination in Sentiment Analysis / Opinion Mining at the sentence level.
By well equipped we mean for it to be capable of delivering same or better results than the most commonly used techniques whilst staying closer to the utilisation of models that are a good match to deal with problems related to ambiguity and uncertainty, as the latter are present in natural languages. For simplicity, this question can be decomposed into three sub-questions:

1. Are *lexicon-based methods* capable of delivering similar *precision* to the one provided by *Supervised Machine Learning* techniques in the determination of polarity subjectivity in Sentiment Analysis?

2. Are *fuzzy methods* adequate to support subjectivity determination and model polarity in Sentiment Analysis by introducing gradualness (graduality) represented through the application of fuzzy sets?

3. Are *semantic rules* a good mechanism for computing semantic orientation in both, words and sentences?

Is there going to be synergy among all these elements? Currently, most of research performed has been conducted using Supervised Methods in Machine Learning (mostly SVM, Naïve Bayes and others). Hence, our comparing base will be defined by some of the latter methods.

In a way, we must try to determine whether obtaining good results in these three sub-questions will have an aggregated *positive* effect when all get combined together. The key performance indicators that will be chosen for the comparison will be decisive in understanding how successful our journey has been.

As a key clarification, let us mention that in our research, the focus will be at the *Sentence Level* - Sentiment Analysis. As such, as expressed by Liu [131] "*Assumption of sentence-level sentiment classification*: The sentence expresses a single opinion from a single opinion holder". In this case, notice that the quintuple $(e_j, a_{jk}, so_{ijkl}, h_i, t_l)$ 2.1 is not utilised to address the problem [131].

In closing, we present below a diagram depicting a high-level view of a possible hybrid solution focused specifically at the Sentence Level and utilising lexicon-based methods (Figure 5.1). Notice that the direct input represented by the 'Stream of Characters' input-box in the diagram, corresponds to a set of sentences with emoticons removed (in the case of the Twitter dataset). The sentences are just plain *character strings* ended by a line-feed and carriage-return each. Spelling errors, special characters and numbers are all present. Notice that the pre-processing occurs in the next step in the diagram, as shown by the 'Pre-processing & PoS tagging' box. In the figure, the two greyed areas correspond to the fields that we are focusing on in our research:

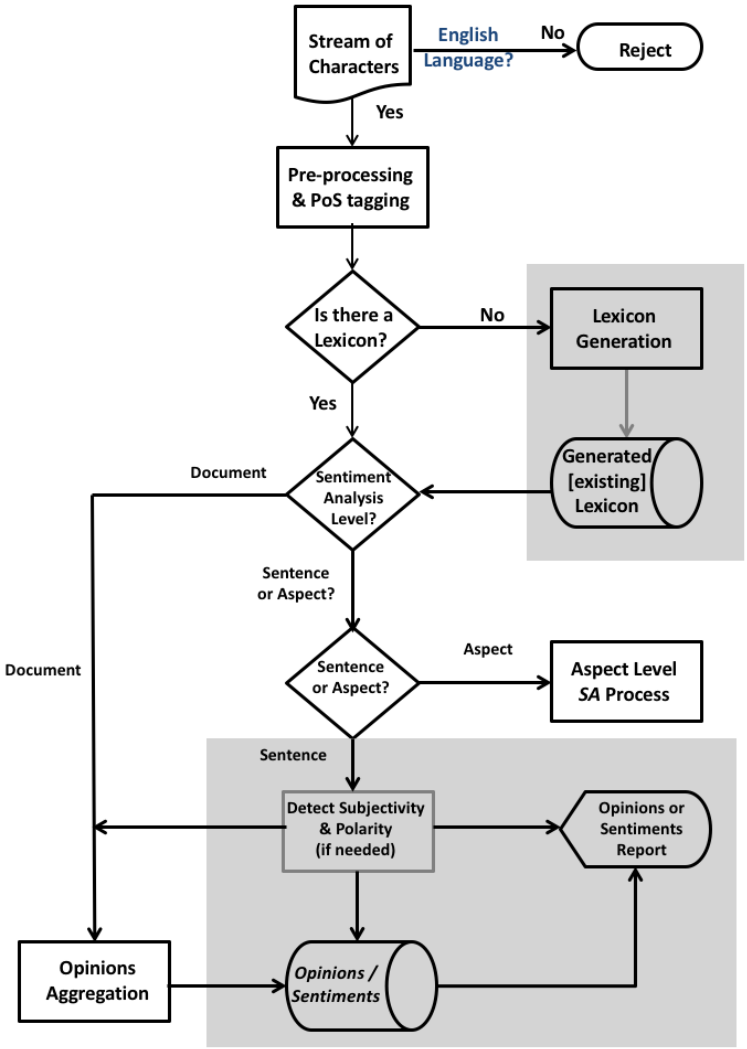


Fig. 5.1. Generic view of a possible lexicon-based solution addressing SA at the sentence level

1. *Building a sentiment lexicon*: the creation of a sentiment lexicon, counting with sentiment-conveying terms (words), part-of-speech tags and polarity scores for each term (see Section 14.1.1).

2. *Devise the necessary logic to evaluate the polarity of sentences (by using a sentiment lexicon-based approach)*: the devising of the algorithms that will produce a classification output using a number of rules and the sentiment lexicon previously defined (see Sections 14.1.2 and 14.1.2.1).

3. *Design a process to identify the intensity/graduality of sentences*: the generation of a mechanism to identify the intensity/graduality of a given sentence (see Section 14.1.3).

The rest of elements in Fig 5.1 are constituted by:

1. A decision point securing that the language of the input is *English*.

2. A decision point that ensures that there is a sentiment lexicon available already.

3. A Pre-processing & part-of-speech tagging step (see Chapter 7).

4. A decision point making sure that we can focus on sentiment analysis at the sentence level (notice that any other type, *document* or *aspect* levels, are rejected in this diagram.

There are other aspects that contributed to our motivation and hypothesis-making. Dzogang et al. stated in [74] that usually authors refer mainly to psychological models when attacking the SA problem. However, other models may be successful as well in this domain. As per Dzogang et al. "it must be underlined that some appraisal based approaches make use of graduality through fuzzy inference and fuzzy aggregation for processing affective mechanisms ambiguity and imprecision." When dealing with SA, Bing Liu [133], one of the main world experts in this area, says that "we probably relied too much on Machine Learning". When it comes to discussing the progress in the SA discipline, Poria et al. [176] introduced a novel idea to concept-level sentiment analysis, which involves combining together linguistics, common-sense computing, and machine learning, aiming to improve precision on polarity detection. This approach of merging techniques is as well a *hybrid style* of compounding the power of several tools.

Considering all of the arguments above, we believe that the following concepts could be applied in combination:

- The concept of *graduality* expressed through fuzzy sets.

- The idea that other tools, together, besides Supervised Machine Learning in isolation, may be viable as well when extracting sentiment from text (especially, if combined with other techniques).

- The positive contribution that NLP tools, semantic rules and a solid opinion lexicon can have in identifying polarity.

Based on these arguments, our research hypothesis can be stated as follows:

Hypothesis 1. *A sentiment analysis method at the sentence level, using a combination of sentiment lexicons, NLP essential tools and fuzzy sets techniques, should perform same or better than today's accepted text classification supervised machine learning algorithms when the latter are utilised in isolation.*

We are establishing the aforementioned hypothesis as we are in search of a sentiment analysis method that closely resembles the way human beings deal with this topic. We expect in the future to be able to expand our

method to deal with human-aspects like humour, irony and sarcasm, which most likely will require providing context. However, it is our belief that the sooner we get closer to the way humans process sentiment, the better positioned we will be to take the next step. We call our proposed system a *hybrid*, because of the fact that it uses a combination of methods and techniques that stem from different research disciplines: fuzzy set theory, natural language processing algorithms and linguistic systems.

5.3 Chapter Summary

In this chapter we have addressed our *research hypothesis*. In the next chapter we will expand on the research methodology that was followed.

Chapter 6

Research Methodology

Confusion will be my epitaph,
As I crawl a cracked and broken path,
If we make it we can all sit back and laugh,
But I fear tomorrow I'll be crying,
Yes I fear tomorrow I'll be crying,
Yes I fear tomorrow I'll be crying.

Peter John Sinfield (King Crimson), 1969

A significant part of the content of this chapter was used in articles published by the author. See references [10, 11, 12, 13, 14, 15, 16, 17, 18]. In this chapter we will describe in detail our proposed solution to the sentiment analysis problem at the sentence level. The research methodology that will be used is discussed from three different perspectives: the process that will be followed, the data that will be used and the description of the indicators that will be utilised for measuring the performance of the proposed SA solution.

6.1 The process

In order to measure success, the proposed method should perform *the same or better* than today's accepted supervised machine learning text classification solutions when utilised in isolation. In the specific case of the SA problem, the proposed solution is compared against two supervised machine learning methods that enjoy a high level of acceptance and credibility in the text classification research community and that are relatively easy to implement: Naïve Bayes (NB) (Section 9.1) and Maximum Entropy (ME) (Section 9.2). At this time, we have opted not to compare results against a popular classification technique, Support Vector Machine (SVM), because we are focusing our research hypothesis at performing sentiment analysis at the sentence level, and research from Wang and Manning [225] demonstrated that Naïve Bayes actually outperforms SVMs for 'snippets': "for short snippet sentiment tasks, NB actually does better than SVMs (while for longer documents the opposite result holds)." The comparison will focus on sentiment/opinion polarity determination.

6.2 The data

A natural question to answer at this point is what data to use to benchmark our results. The following subsections will describe the details of the data sets utilised in this study.

6.2.1 Twitter datasets

Based on the terms and conditions for the utilisation of the data and because of privacy acts' related regulations, many Twitter datasets have been withdrawn from public access as a request from Twitter. However, despite the fact just mentioned, there are still a few Twitter datasets available publicly. We have chosen two of them:

- The first one we have utilised is *Sentiment140*. This dataset offers Twitter corpus data available at their site in CSV format and any trace of emoticons has been removed. We will call this data set *Twitter A*. It is available at:
  `http://help.sentiment140.com/for-students`.
- The second dataset has been provided by Twitter Sentiment Analysis Training Corpus. It contains approximately a million and a half classified tweets, each row is marked as 1 for positive sentiment and 0 for negative sentiment. The dataset is based on data from two sources: University of Michigan Sentiment Analysis competition on Kaggle and the Twitter Sentiment Corpus by Niek Sanders. We have randomly chosen 1000 tweets of each type (negative and positive) that have been used in our experiments. We will call this dataset *Twitter B*. This dataset required much more cleansing effort when compared to the Twitter A data (there were numerous errors, mistypes, emoticons, strange characters, etc., that needed to be removed). It can be downloaded at:
  `http://thinknook.com/twitter-sentiment-analysis-training-corpus-dataset-2012-09-22/`.

6.2.2 Movie Review dataset

Pang and Lee [164] published the datasets that were utilised in SA experiments and for which the results were addressed and discussed in [162, 163, 165]. The datasets are sub-divided into categories, namely, sentiment polarity datasets, sentiment scale datasets and subjectivity datasets. As such, it seems adequate to use the Movie Review Dataset provided by Pang and Lee. The fact that many articles in SA discuss this dataset and have used it to validate their own methods and approaches makes it an ideal candidate from the benchmarking angle. This dataset was first used in the experiments described in [163]. The dataset is available at:
`http://www.cs.cornell.edu/people/pabo/movie-review-data/`.

6.3 Indicators in the evaluation of Sentiment Analysis

It has become customary to evaluate the performance of sentiment classification systems utilising the following four indeces, as defined in [91, 189] (refer to the so-called *confusion matrix* given in Table 6.1):

- *Accuracy* – the portion of all true predicted instances against all predicted instances:

$$\frac{TP+TN}{TP+TN+FP+FN}$$

- *Precision* – the portion of true positive predicted instances against all positive predicted instances:

$$\frac{TP}{TP+FP}$$

- *Recall* – the portion of true positive predicted instances against all actual positive instances:

$$\frac{TP}{TP+FN}$$

- *F1-score* – a harmonic average of precision and recall:

$$\frac{2\times Precision\times Recall}{Precision+Recall}$$

	Predicted Positives	**Predicted Negatives**
Actual Positive instances	# of True Positive (TP) instances	# of False Negative (FN) instances
Actual Negative instances	# of False Positive (FP) instances	# of True Negative (TN) instances

Table 6.1. Confusion Matrix

Readers are referred to the work by Sadegh et al. [91, 189] for more elaborated details on these performance indicators.

6.4 Chapter Summary

The research methodology we have followed has been described in this chapter. *Part III* on this report, covering *methods, techniques and tools*, will be developed next.

Part III

METHODS, TECHNIQUES & TOOLS

Chapter 7

A summary of the mechanics of Text Manipulation

> "When Odin arrived, he asked Mimir for a drink from the water. The well's guardian, knowing the value of such a draught, refused unless the seeker offered an eye in return. Odin -whether straight away or after anguished deliberation, we can only wonder- gouged out one of his eyes and dropped it into the well. Having made the necessary sacrifice, Mimir dipped his horn into the well and offered the now-one-eyed god a drink."
>
> Paul C. Bauschatz, 1982. The Well and the Tree: World and Time in Early Germanic Culture.

This chapter presents a summary of the main techniques that are used in text manipulation, as the latter plays a significant role in the data preparation stage that precedes sentiment classification. As in other disciplines, the pre-processing of the data is vital in the domain of SA. If we recall the material covered in subsection 2.2.3 (see Figure 2.2), we see that the *tokenization phase* and some *Lexical Analysis tasks* are performed before any further analysis can take place. Fundamentally, before we move into a proper analysis the text must somehow be prepared. We need firstly to remove noise, parse the paragraphs and generate proper Part-Of-Speech (POS) tags as a pre-work step.

7.1 Sentences as Unstructured Data

Natural Language Processing (NLP) is a field of computer science, artificial intelligence, and linguistics concerned with the interactions between computers and human languages ([3] and `http://en.wikipedia.org/wiki/Natural_language_processing`). *Sentences* are usually presented as text (string of characters) and *documents* can de described as a collection of sentences. For many, text -as the vehicle used to represent Natural Languages- is the ultimate Unstructured Data. In a way, before any analysis can be performed, data -text- must be organised and structured in order to be able to make sense out of it.

7.2 Bag of Words (BoW)

A Bag of Words is a representation used in NLP and Information Retrieval (IR) to address the unstructured nature of sentences in paragraphs in documents. A bag is a set that allows repetition in its members. Let us use an example to visualise a BoW. The example below was inspired after `http://en.wikipedia.org/wiki/Bag-of-words_model`. Let us look at two sentences: *Document*1 = "Robert likes to watch football. His girlfriend likes football too" and *Document*2 = "Robert also likes to watch movies". Let us build a dictionary based on these two sentences:

{
“Robert”: 1,
“likes”: 2,
“to”: 3,
“watch”: 4,
“football”: 5,
“his”: 6,
“also”: 7,
“movies”, 8,
“games”: 9,
“girlfriend”: 10,
“too”: 11,
“romantic”: 12
}

There are 12 different words represented in the dictionary. As there are 12 different unrepeated words in the BoW, a vector with 12 entries can be used to represent the contents of *Document* 1 and *Document* 2. The number that follows every distinct word -in the dictionary- corresponds to the position used to represent a given word in the vector. E.g. in the first vector below, the fifth position represents the word ”football”.

[1, 2, 1, 1, 2, 1, 0, 0, 0, 1, 1, 0] representing *Document*1
[1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0] representing *Document*2

The number in the vector, represents how many times that word appears in the Document. For the first vector representing *Document* 1 the word “football” shows 2 times and the word “too” only once.

7.3 Tokenization

Once a sentence -or a full document- is provided, it is entered as a string of characters. Somehow, this string must be broken into pieces that allow a more effective manipulation. This string contains items that are noise or are of no interest at all, like white-space, blank-lines, line-breaks, carriage-returns, etc. *Tokenization* is the process of taking a string of characters of length n as input and breaking it up into words and punctuation as an output [25, 26]. Let us look at an example.

Let us tokenize some text, using an example taken from [26]:

```
>>> import nltk
>>> sentence = """At eight o'clock on Thursday morning Arthur didn't feel very good."""
>>> tokens = nltk.word_tokenize(sentence)
>>> tokens
['At', 'eight', "o'clock", 'on', 'Thursday', 'morning', 'Arthur', 'did', "n't", 'feel', 'very', 'good', '.']
```

The output above has been obtained using the Programming Language Python and NLTK (Natural Language Toolkit), a set of libraries and code for Natural Language Processing [25]. If we analyse the answer stored in the variable *tokens*, we notices that all tokens are words or punctuation. Of particular importance is the way a verb in a negation form (didn't) has been treated. It is presented as two tokens, respectively $'did'$ and $"n't"$. *Tokenization* is the beginning of this manipulation process, but there is much more to it. In Appendix D, we provide more information on tokenization and data manipulation & processing in general. In the aforementioned appendix we include as well information on the specific tokenizer and parser that were utilised to process sentences in the experimental datasets, as a preamble to the actual sentiment classification process.

7.4 POS Tagging

In [156], Nau uses the above example to introduce the topic of Part Of Speech (POS) Tagging. In a given sentence in the English language we can find words that may belong into many of the different established identifiable POS particles. As per [156], let us analyse the sentence "Flies like a flower":

- *Flies*: noun or verb?
- *like*: preposition, adverb,conjunction, noun or verb?
- *a*: article, noun, or preposition?
- *flower*: noun or verb?

This situation is known as lexical ambiguity and it can be resolved with a technique known as Part-of-Speech *tagging*. The POS tagger is usually executed *before* a parse tree (Section 7.5) is built. Let us see a couple of examples taken from [26]. We will use the example presented in subsection 7.3 (we will continue using the variable *tokens* that stores the tokenization version of the example sentence.

```
>>> tagged = nltk.pos_tag(tokens)
>>> tagged [ 0 : 6 ]
[('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'), ('on', 'IN'), ('Thursday', 'NNP'), ('morning', 'NN'),
('Arthur','NNP'), ('did','VBD'), ("n't",'RB'), ('feel','VB'), ('good','JJ'), ('.','.')]
```

If we look with attention to this output and inspect the contents of variable *tagged*, we noticed that labels belonging in the set of established POS have been assigned to the example sentence. It seems that the default tagger for the version of NLTK we are using is a Maximum Entropy tagger trained on the Penn Treebank corpus [141]. The meaning of the PoS-labels displayed in the example shown above is covered in Table 7.1.

Label	Description
IN	Preposition/Subordinating conjunction
CD	Cardinal Number
JJ	Adjective
NNP	Proper Noun, Singular
NN	Noun, Singular or mass
VBD	Verb, Past tense
RB	Adverb
VB	Verb, Base form
.	Sentence-final punctuation

Table 7.1. Labels used by the POS tagger in the example shown

There are a number of methods that are used for POS tagging. As explained by Nau [156]:

- Rule-Based POS Tagging
- Transformation-Based Tagging (e.g. Brill's tagger [36])
- Stochastic Tagging

POS tagging is supposed to be very useful for many reasons. Especially for Word-Sense disambiguation and to reduce the number of parses required. Most of modern POS-taggers correspond to one of the methods described above, or to a combination of them. See Appendix D for more details on the PoS tagger we utilised in our work, that according to Bird [25, 26] is very efficient.

7.5 Parsing

Once the POS tagger has completed its job, we still need to move forward and generate a parse-tree and identify named entities. For continuity, let us proceed with the same example we have mentioned before. Identify named entities and parse-tree. As taken from [26] :

```
>>> entities = nltk.chunk.ne_chunk(tagged)
>>> entities
Tree('S', [('At','IN'), ('eight','CD'), ("o'clock",'JJ'),
           ('on', 'IN'), ('Thursday', 'NNP'), ('morning', 'NN'),
      Tree('PERSON', [('Arthur','NNP')]),
           ('did', 'VBD'), ("n't",'RB'), ('feel', 'VB'),
           ('very', 'RB'), ('good', 'JJ'), ('.', '.')])
```

Notice that the full parse-tree includes a tree inside the original tree (a sentence inside a sentence) and a named-entity, a 'PERSON' called 'Arthur' has been identified. As we continue with more complex analysis tasks, this type of information would prove to be very useful, as it will assist on identifying the different elements presented in an opinionated sentence (multiple nested *subjects*, scope of *negation*, etc.).

7.6 Lemmatisation & Stopwords

A lemma is the *canonical* form of a given word and the process of determining the lemma for a given word is referred to as lemmatisation. For example, the chosen form of a noun is the singular ('horse' instead of 'horses'). For the verbs, 'run, runs, ran and running' are represented by the word 'run', that is the associated lemma (sources: [183] and Wikipedia). As we can see using *lemmas* is very useful to simplify the SA process. We can refer to the *lemma* of a given word instead of worrying about all different forms associated to a word (note: for other activities in NLP, like obtaining full semantics out of a phrase, differentiating between a singular or a plural in a noun may be critical). For instance, for the creation of a dictionary to be used in SA, the words in the aforementioned dictionary will be indexed by the lemma.

What about stop words (or stopwords)? In computational linguistics and NLP stop words are words that are carved out or removed out before doing the analysis of the *input data*. The list of stopwords will change depending on the application. The needs of filtering out some words in a search engine (elimination of articles, etc.) are different than those associated to the SA problem and the level of detail used in a SA application. For example, stopwords in text coming from documents or reviews will have to be approached differently than when the input text come from Twitter. For instance, NLTK [25] comes with a so-called *stopwords corpus*, that contains 128 stopwords for the English language. The NLTK documentation provides all required details on the subject (`http://www.nltk.org/` and `https://github.com/nltk/nltk/wiki`). As expected, it is possible to define your own stopwords list. In our particular case, the analysis is focused on sentences. As such, we have used the *stopwords list* embedded in our NLTK [26] package, which represents a best-practices approach to the problem of NLP data pre-processing (see Appendix D).

7.7 Chapter Summary

In this chapter we briefly touched base on the main aspects of the mechanics of text manipulation. These tasks are an essential preamble to the core tasks of natural language processing, and more specifically, to the SA discipline as a whole. In the next chapter we will introduce the basics of *machine learning* and *lexicon-based approaches* to the SA problem.

Chapter 8

Machine Learning & Lexicon-Based Approaches to SA

> The earliest known dictionaries were kept in the Mesopotamian city of Elba (now part of Syria). These clay tablets inscribed in columns of cuneiform writing date from about the 2300s BC and consist of words in the Sumerian language and their equivalents in the Akkadian language.
>
> Dictionary - MSN Encarta, 2009.

In this chapter we briefly discuss the two main approaches to SA. This is relevant because in our proposed method (see Part IV) we have selected one of these aforementioned approaches as our main technique.

As discussed in Section 2, there are two recognised main approaches to the problem of extracting sentiment automatically from a given source.

(a) The lexicon-based approach, that involves calculating orientation for a document/sentence from the semantic orientation of words or phrases in the source.

(b) The classification approach which involves building classifiers from labeled instances of texts or sentences.

Taboada et al. [208] discuss the most common and accepted approaches to extract sentiment automatically from texts. In Figure 8.1 we have attempted to summarise those, as we will discuss the subject further down the road. A third option to address the SA problem is the (c) rule-based approach, which basically "looks for opinion words in a text and then classifies it based on a number of positive and negative words. It considers different rules for classification such as dictionary polarity, negation words, booster words, idioms, emoticons, mixed opinions, etc." [64]. A domain-independent rule based system for semantically classifying sentiment from customer reviews [115] is a good example of this approach. A fourth option could be the so-called (d) Statistical model approach, where each review is represented "as a mixture of latent aspects and ratings. It is assumed that aspects and their ratings can be represented by multinomial distributions and try to cluster head terms into aspects and sentiments into ratings" [64]. A good representative of this approach is the work of Moghaddam and Ester [152]. Both options (a) and (b) are used extensively. During our research, we have developed a *hybrid approach* that corresponds to a lexicon-based model that incorporates semantic rules and fuzzy set theory to model graduality (see Chapter 14). In summary, we have devised a model based on option (a) and will compare it against two well-known algorithms that represent option (b).

8.1 ML Approaches

We will review in more detail the techniques associated to Machine Learning (ML) in Chapter 9 (Supervisd ML) and Chapter 10 (Unsupervised ML). Tom Mitchell's definition of ML follows: [151] "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E". As already mentioned, the fundamental idea behind ML approaches is

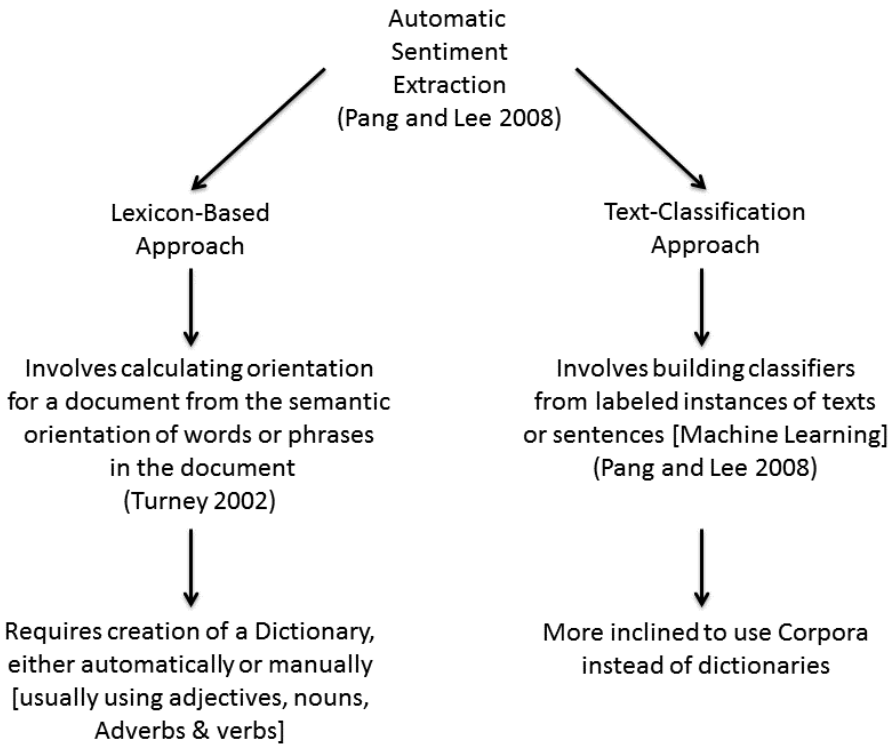


Fig. 8.1. Automatic extraction of sentiment, as per reproduced from Taboda et al. [208]

the utilisation of a number of learning algorithms that are applied to datasets that have been previously labelled, so the patterns/features akin to the cases are learnt. Then, the new (unseen) data is presented to the algorithms, which will attempt to classify the new datasets, accordingly. ML has strong connections to the worlds of Statistics and Optimisation, as well as to theoretical Computer Science. For more details on ML, please refer to [8, 27, 72, 114, 121, 124, 134, 142, 151].

8.2 Lexicon-Based Methods (LBM)

The fundamental idea behind a Lexicon-based method is the use of a dictionary of sentiment words and even phrases. For our discussion, we will focus in the paper by Taboada et al. [208]. Let us keep in mind that Semantic Orientation (SO) is defined as a measure of subjectivity and opinion in text. "sentiment analysis refers to the general method to extract subjectivity and polarity from text (potentially also speech), and semantic orientation refers to the polarity and strength of words, phrases, or texts. Our concern is primarily with the semantic orientation of texts, but we extract the sentiment of words and phrases towards that goal" [208]. We would like to stress that in lexicon-based approaches, dictionaries of words annotated with the word's semantic orientation, or polarity, are required. In some cases, lexicon-based methods utilise unsupervised learning methods, of which [212, 213, 214] are very good examples, or focus in the utilisation of Semantic Rules and a number of NLP techniques. In Part IV of this thesis the lexicon based approach will be fully described and a new hybrid Lexicon-based Method to address the SA problem will be developed and experimentally compared against some of the most used SML approaches in SA.

8.3 Chapter Summary

In this chapter we have addressed the machine learning and the lexicon-based approaches to the SA problem. We have as well refer the reader to the chapters in this document where we put at use these methods. In the next chapter we will cover Supervised Machine Learning methods.

Chapter 9

Supervised Machine Learning (SML)

> "Those who educate children well are more to be honoured than they who produce them; for these only gave them life, those the art of living well."
>
> Aristotle.

This chapter reviews supervised learning techniques with a view to adopting some of them as the basis for performance comparison against our proposed method, which is introduced in Part IV. There are plenty of tested and mature algorithms for text classification that rely on supervised machine learning. According to Alpaydin [8], In Supervised Machine Learning, the fundamental idea is to learn a mapping from the input to an output. The correct values for this output are provided by a supervisor. Hence, supervised algorithms make use of labelled training data to create a function, where this aforementioned function is supposed to predict the correct output. Generally speaking [142], "The classification problem consists of taking input vectors and deciding which of N classes they belong to, based on training from *exemplars* of each class". The assumption for this approach is that each member of the *exemplars* set belongs to one and only one class. In addition, the set of classes encompasses all possible answers in the solution space. These assumptions collide with our understanding of the real world, but it seems that they enable some of these algorithms to behave with certain level of accuracy.

In this Chapter, we focus on three supervised machine learning methods, namely, Naïve Bayes (NB), Maximum Entropy (ME) and Support Vector Machine (SVM), as these three techniques are largely utilised and are considered to be among the best in their class.

9.1 Naïve Bayes (NB)

As a preamble, we will use some of the definitions provided in [164, 165] for the Machine Learning problem. Pang et al. [165] argue that "Our aim in this work was to examine whether it suffices to treat sentiment classification simply as a special case of topic-based categorization (with the two'topics' being positive sentiment and negative sentiment), or whether special sentiment-categorization methods need to be developed". Then they proceed defining what they call a *standard bag-of-features framework*. This is formally represented as [165]: Let $\{f_1, \ldots, f_m\}$ be a predefined set of m *features* that can appear in a document. Let $n_i(d)$ be the number of times f_i occurs in document d. Then each document d is represented by the document vector $\vec{d} := (n_1(d), n_2(d), \ldots, n_m(d))$.

In discussing the NB technique, Pang et al. [165] elaborate that one possible approach to text classification is to assign to a given document d the class $c^* = arg\,max_c\, P(c \mid d)$. Before continuing, let us recall the Bayes' rule:

$$P(c \mid d) = \frac{P(c)\, P(d \mid c)}{P(d)}, \tag{9.1}$$

where $P(d)$ has no role at all in selecting c^*. According to [165], "To estimate the term $P(d\,|\,c)$, Naive Bayes decomposes it by assuming the f_i's are conditionally independent given d's class":

$$P_{NB}(c\,|\,d) := \frac{P(c)\,(\prod_{i=1}^{m} P(f_i\,|\,c)^{n_i(d)})}{P(d)}. \tag{9.2}$$

Pang et al. continues by saying that the training method they used "consists of relative-frequency estimation of $P(c)$ and $P(f_i\,|\,c)$, using add-one smoothing". In essence, NB is a simple probabilistic classifier that is based on the Bayes' theorem. Basically, in the presence of a sample input NB should be able to predict a probability distribution over a set of classes. In this case, word frequencies are the characteristics used to decide whether a paragraph belongs to one category or another one. For that to happen, we would have to count with a dictionary (or corpus) previously labelled with the semantic orientation of words (i.e. 'fabulous' conveys a positive intention whilst 'terrible' would convey a negative one).

Despite its apparent simplicity, NB has proven to be very successful in many situations. Quoting [165], "Despite its simplicity and the fact that its conditional independence assumption clearly does not hold in real-world situations, Naïve Bayes-based text categorization still tends to perform surprisingly well". We have shown above the formal definition of the NB classification method. Now, we will show a description of the NB algorithm more centred around SA and more didactic, too. It is based on a *Sentiment Symposium Tutorial* given by Christopher Potts from the Center for the Study of Language and Information from Stanford University (San Francisco, November 2011) [179]. Please see URL: `sentiment.christopherpotts.net/index.html`.

As mentioned already, the NB method has the reputation to be the simplest probabilistic classifier model based on data used previously for training. According to Potts [179] the following (verbatim) is a recipe for training an NB classifier (using just the words as features):

1. Estimate the probability $P(c)$ of each class $c \in C$ by dividing the number of words in documents in c by the total number of words in the corpus.
2. Estimate the probability distribution $P(w\,|\,c)$ for all words w and classes c. This can be done by dividing the number of tokens of w in documents in c by the total number of words in c.
3. To score a document d for class c, calculate

$$\textbf{score}(d,c) \stackrel{\text{def}}{=} P(c) * \prod_{i=1}^{n} P(w_i\,|\,c)$$

4. If you simply want to predict the most likely class label, then you can just pick the c with the highest score value. To get a probability distribution, calculate

$$P(c\,|\,d) \stackrel{\text{def}}{=} \frac{\textbf{score}(d,c)}{\sum_{c' \in C} \textbf{score}(d,c')}$$

Potts continues writing [179] "The last step is important but often overlooked. The model predicts a full distribution over classes. Where the task is to predict a single label, one chooses the label with the highest probability. It should be recognized, though, that this means losing a lot of structure. For example, where the max label only narrowly beats the runner-up, we might want to know that. The main drawback to the NB model is that it assumes each feature to be independent of all other features. This is the 'naïve' assumption seen in the multiplication of $P(w_i\,|\,c)$ in the definition of **score**. Thus, for example, if you had a feature best and another world's best, then their probabilities would be multiplied as though independent, even though the two are overlapping. The same issues arise for words that are highly correlated with other words (idioms, common titles, etc.)".

9.1.1 Naïve Bayes Explained

In simple terms, the NB algorithm enables us to predict a class, given a set of features using probability. AYLIEN [19], in their BLOG on Data Science (`http://blog.aylien.com/naive-bayes-for-dummies-a-simple-explanation/`),

provides a simple explanation and example to aid in understanding how Naïve Bayes work. For the sake of the argument, let us assume that we have data on 1,000 pieces of fruit. The fruits in the dataset are bananas, oranges and some other fruit. In order to *describe* the fruits we rely on three features of the fruits as described in Table 9.1 below. We could

Fruit	**Long**	**Sweet**	**Yellow**	**Total**
Banana	400	350	450	500
Orange	0	150	300	300
Other	100	150	50	200
Total	**500**	**650**	**800**	**1000**

Table 9.1. Fruits Example - Features & Quantities, as presented in [19]

predict whether a fruit is a banana or an orange, based on its features; namely, long-shape, sweetness and colour. In terms of distribution, in the complete set of fruits the percentage allocation is: 50% of the fruits are bananas, 30% are oranges and 20% are other fruits. From our training set, which characteristics are displayed in Table 9.1, we can assert that, from 500 bananas:

- 400 are *long*.
- 350 are *sweet*.
- 450 are *yellow*.

Likewise, from 300 oranges, 0 are *long*, 150 are *sweet* and 300 are *yellow*. From the remaining 200 fruits, 100 are *long*, 150 are *sweet* and 50 are *yellow*. Do we have enough evidence to predict the class to which another fruit belongs? Let us assume that we receive as input for the next fruit, the following features: *long, sweet and yellow*, then we can classify it using the Naïve Bayes Equations 9.1 or 9.2 for each case and compare all of the results. The winner will be the one with the highest probability, as follows:

- P(Banana) = 0.252 .
- P(Orange) = 0 .
- P(Other fruit) = 0.01875 .

The highest probability is 0.252, so as per Naïve Bayes we can assume that the *long, sweet and yellow* fruit that was recently introduced is in fact a Banana.

9.2 Maximum Entropy (EM)

The Maximum Entropy classification algorithm (EM or MaxEnt) is another technique that has been extensively used by the Machine Learning community to deal with NLP problems. According to [165], in some situations it may outperform Naïve Bayes in text classification tasks. ME estimate of $P(c\,|\,d)$ is expressed as follows:

$$P_{ME}(c\,|\,d) \;:=\; \frac{1}{Z(d)}\,\exp\left(\sum_i \lambda_{i,c}\,F_{i,c}(d,c)\right) \tag{9.3}$$

where $Z(d)$ is a normalization function, $F_{i,c}$ is a *feature/class function* for feature f_i and class c, defined as follows:

$$F_{i,c}(d,c') := \begin{cases} 1 & n_i(d) > 0 \text{ and } c' = c \\ 0 & \text{otherwise} \end{cases} \tag{9.4}$$

One of the main properties of EM is that the algorithm does not make any assumptions about the relationships between features. The $\lambda_{i,c}$'s are feature-weight parameters. If we look at the definition of P_{ME} we notice that a large value for $\lambda_{i,c}$ would imply that "f_i is considered a strong indicator for class c. The parameter values are set so as to maximize the entropy of the induced distribution subject to the constraint that the expected values of the feature/class functions with respect to the training data: the underlying philosophy is that we should choose the model making the fewest assumptions

about the data while still remaining consistent with it, which makes intuitive sense" [165]. Additional information about EM can be found in the literature, like [8, 27, 142] and others.

9.2.1 Maximum Entropy Explained

In their article [158], Nigam et al. explain that the motivation behind maximum entropy is that "one should prefer the most uniform models that also satisfy any given constraints". For example, assume that we are in the presence of a four-way text classification task. The only piece of information we are given is that on average, 40% of documents with the word 'professor' in them are in the faculty class. Then, when we are given a document with the particle 'professor' in it, we can say that it has a 40% chance of being a faculty document; we can say as well that it has a 20% chance to belong in each one of the other three classes. If a document does not have the word 'professor' we would guess then the uniform class distribution, 25% each. In essence, it is a classifier which rather select the uniformity (maximum entropy) if no data is observed. As the algorithm starts *seeing* the data, it is forced to take distance from the maximum entropy by *explaining data*. Once the data has been explained, the classifier tries again to maximize the entropy on whatever data left that it has not examined yet.

9.3 Support Vector Machine (SVM)

The Support Vector Machine (SVM) algorithm is attributed to Vladimir Vapnik, as part of the material covered in his books [222, 223]. However, it seems that the original idea goes back to work carried out by Vapnik in the mid-seventies and published in Russian in 1979 [221]. For this section we have relied heavily on the work of Burges [39] and Vapnik himself [222]. As mentioned by many authors, among them [130, 132, 133, 164, 165], SVM has been used successfully in classification and regression analysis.

As discussed in [121, 142], SVM is a classifier based on statistical learning. SVM is a non-probabilistic classifier. Informally speaking, SVM operates by constructing a model that assigns occurrences of data that it has never seen before to one category or another, *provided that* previously the SVM learning algorithm has been given a set of training examples (those examples have been previously labelled as belonging to one of the two categories being considered). Basically, SVM analyses datasets and identifies patterns in the data [2]. In order to define SVM properly we must go back to the concept of linearly separable binary classification. Given L training points, where each input x_i has D attributes and is in one of the two classes $y_i = -1$ or $y_i = +1$, the training data is of the form

$$\{x_i, y_i\} \; where \; i = 1 \ldots L, \; y_i \in \{-1, 1\}, \; x \in \Re^D$$

We are working under the assumption that the data is linearly separable (we can draw a line in a graph separating both classes of objects when $D = 2$ and a *hyperplane* on graphs x_1, $x_2 \ldots x_D$, when $D > 2$). This hyperplane can be described by $w \cdot x + b = 0$ where [85]:

- w is normal to the hyperplane.
- $\frac{b}{\|w\|}$ is the perpendicular distance from the hyperplane to the origin.

"Support Vectors are the examples closest to the separating hyperplane and the aim of Support Vector Machines is to orientate this hyperplane in such a way as to be as far as possible from the closest members of both classes." [85].

If we analyse Figure 9.1 -reproduced from the original graphic presented in [85]-, "implementing a SVM boils down to selecting the variables w and b so that our training data can be described by":

$$x_i \cdot w + b \geq +1 \quad for \; y_i = +1 \tag{9.5}$$

$$x_i \cdot w + b \geq -1 \quad for \; y_i = -1 \tag{9.6}$$

Combining these two inequalities we get:

$$y_i(x_i \cdot w + b) - 1 \geq 0 \; \forall_i \tag{9.7}$$

In Figure 9.1, the points that are closest to the separating hyperplane, called *Support Vectors*, are highlighted by being

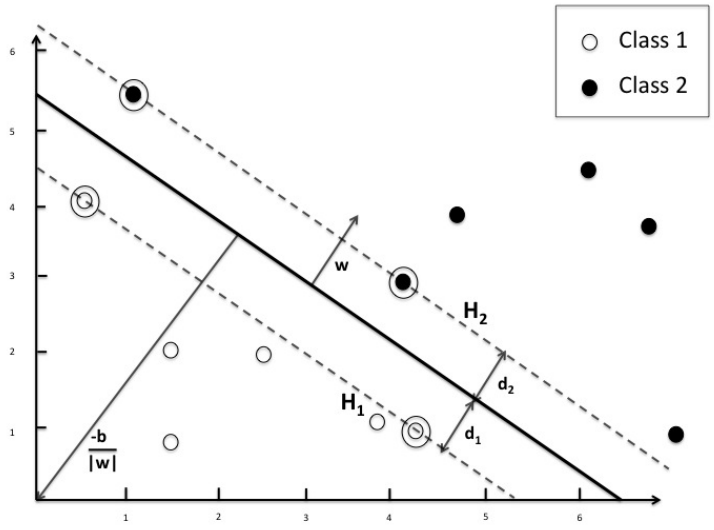


Fig. 9.1. Hyperplane through two linearly separable classes

shown in circles. "The two planes H_1 and H_2 that these points that lie on can be described by" [85]:

$$x_i \cdot w + b = +1 \quad \text{for} \quad H_1 \tag{9.8}$$

$$x_i \cdot w + b = -1 \quad \text{for} \quad H_2 \tag{9.9}$$

In Figure 9.1, d_1 is defined as the distance from H_1 to the hyperplane and d_2 as the distance from H_2 to the aforementioned hyperplane. "The hyperplane's equidistance from H_1 and H_2 means that $d_1 = d_2$ (a quantity known as the SVM's margin). In order to orientate the hyperplane to be as far from the Support Vectors as possible, we need to maximize this margin" [85]. A number of algebraic manipulations follow as per [85]. The aforementioned *margin* is equal to $\frac{1}{\|w\|}$. Attempting to maximise this function as per equation 9.7 is equivalent to obtaining:

$$\min \| w \| \quad \text{such that} \quad y_i(x_i \cdot w + b) - 1 \geq 0 \; \forall_i$$

Hence, minimising $\| w \|$ is equivalent to minimising $\frac{1}{2} \| w \|^2$ and "the use of this term makes it possible to perform Quadratic Programming (QP) optimization later on" [85]. Therefore, we must find:

$$\min \frac{1}{2} \| w \|^2 \quad \text{s.t.} \quad y_i(x_i \cdot w + b) - 1 \geq 0 \; \forall_i \tag{9.10}$$

9.3.1 Support Vector Machine Explained

Below, a step-by-step process on how to use an SVM to solve a linearly separable binary classification problem.

- Create **H**, where $H_{ij} = y_i y_j x_i \cdot x_j$
- Find α so that

  $$\sum_{i=1}^{L} \alpha_i - \frac{1}{2} \alpha^T H \alpha$$

  is maximized, subject to the constraints

  $$\alpha_i \geq 0 \; \forall_i \text{ and } \sum_{i=1}^{L} \alpha_i y_i = 0.$$

  This is done using a QP solver.

- Calculate
$$\mathbf{w} = \sum_{i=1}^{L} \alpha_i y_i x_i.$$
- Determine the set of Support Vectors S by finding the indices such that $\alpha_i > 0$.
- Calculate
$$b = \frac{1}{N} \sum_{s \in S} (y_S - \sum_{m \in S} \alpha_m y_m x_m \cdot x_S).$$
- Each new point x' is classified by evaluating
$$y' = sgn(w \cdot x' + b).$$

If the data is not completely linearly separable, Fletcher argues that the SVM methodology can be extended by introducing some manipulations (i.e. relaxing some constraints in 9.5 and 9.6). We will not present the development to make the SVM capable of dealing with not fully linearly separable problems. Instead, we would rather refer the reader to [85].

9.4 Naïve Bayes (NB) Vs. Support Vector Machine (SVM) for snippets

Despite the fact that SVM has become one of the most widely utilised supervised machine learning algorithms -especially for classification- and that it is in general a method that shows higher accuracy and precision results, there are cases where a naïve approach, like the one implements through Naïve Bayes produces better results. At this time, we have opted not to compare our experimental results against SVM, because we are focusing our research hypothesis on performing sentiment analysis at the sentence level, and research from Wang and Manning [225] demonstrated that Naïve Bayes actually outperforms SVMs for 'snippets': "for short snippet sentiment tasks, NB actually does better than SVMs (while for longer documents the opposite result holds)."

Hence, Naïve Bayes and Maximum Entropy, the latter being a method that seems to *slightly* outperform Naïve Bayes under specific circumstances, are the chosen methods for comparison purposes against our proposed solution (see Part IV). The comparison will focus on the precision of the sentiment/opinion polarity determination task.

9.5 Chapter Summary

In this chapter we have discussed the fundamentals of key supervised machine learning methods, like Maximum Entropy (ME) and Naïve Bayes (NB). As we will be comparing our proposed method's results with those obtained when using the aforementioned machine learning methods, the topic possesses some relevance. In the next chapter we will address another variant of machine learning, namely, Unsupervised Machine Learning.

Chapter 10

Unsupervised Machine Learning (UML)

> "Many of the fundamental ideas in artificial intelligence have an ancient heritage. Some of the most fundamental, surely, are that thinking is a computational process, that computational processes involve combining symbols, that computation can be made mechanical, and that the mathematics of computation involves combinatorics. All of these ideas have their origin, so far as we know, in the work of an eccentric 13th century Spanish genius, Ramón Lull (1232-1316). Lull's sources were partly mystical, but the interesting part of his thought drew from -or against- an analytic tradition in logic and combinatorics."
>
> Excerpt from the article *Ramón Lull and the Infidels*, by Clark Glymour, Kenneth M. Ford & Patrick J. Hayes, AI Magazine, Vol. 19, No. 2, pp. 136, 1998.

In Unsupervised Machine Learning (UML) we attempt to find structure in data that is unlabelled. According to Alpaydin [8], in Unsupervised Learning there is only input and there is no supervisor or expert. The objective is to find 'the regularities' in the input. We are trying to find 'a structure' to the input in such a way that some patterns occur more frequently than others. These identified patterns would be able to assist in producing a prediction. Because of the way most Unsupervised Learning algorithms work, it is considered that Unsupervised Learning (UL) is somehow domain-independent (product reviews, blogs, political speeches, etc.). In addition, in the case of NLP, UL methods are independent of the language being used in the text or speech being analysed. When an UL system is exposed to unseen new data, there is no need for previously labelling the new data as the method will not require any labelling in order to work. In this regard, UML methods could be a better fit for situations when there are not plenty of linguistic resources at reach or a design decision has been made with the purpose of avoiding having to manually label data. Notice that many of the currently available labelled linguistics resources have been manually labelled through time.

There are a number of techniques that are considered Unsupervised Machine Learning (UML) methods, among them [142], Clustering (K-Means, Hierarchical Clustering, etc.), Hidden Markov models, Self-Organizing Maps (SOM), the Expectation Maximization Algorithm and others. In this section we will address methods that are mentioned in the literature examined as adequate for addressing the subjectivity and orientation problem in SA. In this chapter we will address in more detail the Pointwise Mutual Information - Information Retrieval (PMI-IR) & the Latent Semantic Analysis (LSA) methods and Vector Space Models (VSMs).

10.1 Pointwise Mutual Information - Information Retrieval (PMI-IR) Method

Let us describe the Pointwise Mutual Information - Information Retrieval (PMI-IR) method attributed to Church & Hanks [61] and fully applied in the context of SA by Turney [212]. As described by Turney [212] and Turney & Littman [213], the first part of this method's process is to extract from the input text *phrases containing adjectives and verbs*. Hatzivassiloglou & Wiebe [93] had previously established that it is possible to predict the semantic orientation of adjectives. Turney [212] provides evidence to suggest that Hatzivassiloglou & Wiebe's claim is not always valid as he argues that the same adjective might have different semantic orientations depending on the context it is used [212]. Indeed, the adjective 'unpredictable' has opposite orientation when used in an automotive review to indicate an 'unpredictable steering' (negative) in comparison to its use in a movie review to highlight its 'unpredictable plot' (positive). Thus, it is necessary for the PMI-IR algorithm to extract a pair of words, with one of them being an adjective or adverb while the second one provides context to determine the semantic orientation. As explained by Turney, a part-of-speech tagger, like Brill [36], is applied to the original input text. The algorithm will extract two consecutive words from the input text if its tags correspond to any of the patterns presented in Table 10.1. The aforementioned tags are displayed In Table 10.2, and are presented in

	First Word	**Second Word**	**Third Word (Not Extracted)**
1.	JJ	NN or NNS	anything
2.	RB, RBR, or RBS	JJ	not NN nor NNS
3.	JJ	JJ	not NN nor NNS
4.	NN or NNS	JJ	not NN nor NNS
5.	RB, RBR, or RBS	VB, VBD, VBN, or VBG	anything

Table 10.1. Patterns of tags for extracting two-word phrases from input text

Santorini [192].

Tags	**Definitions**
JJ	Adjective
NN	Noun, singular or mass
NNS	Noun, plural
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle

Table 10.2. Tags definitions as per Santorini [192]

The Pointwise Mutual Information (PMI) between two words, $word_1$ and $word_2$, represents the amount of information that we obtain about the presence of one of the words when we observe the other [212]. PMI is defined as follows [61]:

$$PMI(word_1, word_2) = \log_2 \left[\frac{p(word_1 \,\&\, word_2)}{p(word_1)\, p(word_2)} \right] \tag{10.1}$$

where $p(word_1 \,\&\, word_2)$ represents the probability that $word_1$ and $word_2$ co-occur. Now, what we are trying to identify is the Semantic Orientation (SO) of a given phrase. Turney [212] defines SO as:

$$SO(phrase) = PMI(phrase, \text{“excellent”}) - PMI(phrase, \text{“poor”}) \tag{10.2}$$

According to Turney, he has chosen the words "excellent" and "poor" because in the review system he was analysing, a *poor* review is represented with one star "*" and an *excellent* review is given five stars "*****". Now, let us go back to

the assumptions and rationale provided by Turney.

In equation 10.2, SO is positive when "*phrase* is more strongly associated with 'excellent' and negative when *phrase* is more strongly associated with 'poor'". Turney uses then the *AltaVista Advanced Search Engine* (`www.altavista.com/sites/search/adv` as it is convenient for his analysis. AltaVista has a NEAR operator that "constrains the search to documents that contain the words within ten words of one another, in either order." Turney claims that research has shown that the operator NEAR works better than the logical AND when one is attempting to measure the strength of semantic association between words.

Turney [212] continues by defining hits(*query*) as the number of hits returned given the query '*query*' to the search engine. Then, SO can be calculated by using equations 10.1 and 10.2, performing some algebraic manipulation and assuming the co-occurrence is interpreted as NEAR:

$$SO(phrase) = \log_2 \left[\frac{hits(phrase\ NEAR\ \text{“excellent”})\ hits(\text{“poor”})}{hits(phrase\ NEAR\ \text{“poor”})\ hits(\text{“excellent”})} \right] \tag{10.3}$$

Here are some clarifications as provided directly by Turney [212]:

1. Equation 10.3 is a log-odds ratio
2. To avoid division by zero, Turney added 0.01 to the hits (0.01 was arbitrarily chosen)
3. *phrase* when both *hits*(*phrase NEAR "excellent"*) AND *hits*(*phrase NEAR "poor"*) were simultaneously less than 4, were skipped (4 was arbitrarily chosen)
4. Turney added the particle "AND (NOT host:epinions)" to every query, to tell AltaVista to exclude the Epinions Web site `www.epinions.com/?sb=1` in its searches (Product Review and Consumer Reports)

The third step in the process is then the calculation of the *average semantic orientation of the phrases* in the analysed reviews. A given review will be labelled as *recommended* if the average is positive, and *not recommended* if the average is negative.

In [131] Bing Lui mentions that Turney manages to achieve classification accuracies that range between 84% and 66% on reviews belonging respectively to the automobile reviews and movie reviews categories. According to Liu [131] the main advantage of the method proposed by Turney [212] is that "it provides a prevailing opinion on an entity, topic or event". However, he argues that the main disadvantages are:

- "It does not give details on what people liked and/or disliked" [131]. For Bing Liu, in a review the person writing her thoughts most likely will link her opinions to a specific entity.
- "It is not easily applicable to non-reviews, e.g. forum and blog postings" [131]. In the case on these forums and blogs, most likely opinions will be issued about more than one entity (and many of those opinions might be comparisons among several entities).

10.2 Vector Space Models (VSM) of Semantics & Pointwise Mutual Information (PMI)

How would our Opinion Lexicon improve if we could find a score reflecting the distance (how near or how far) a given lexicon's word is from a few 'seed words' of a pre-recognised extreme polarity, either negative or positive? Basically, we believe we would be looking for border conditions; a sort of a Max/Min in the polarity range $\in [0, \ldots, 1]$ in $\mathbb{R}$. If so, we could assign a meaningful score to every word in our lexicon that would help us to establish how near or how far from the polarity edges -positive or negative- a word is. Let us start with some mathematical concepts.

As mentioned by Turney [210], "statistical approaches to synonym recognition are based on co-occurrence", as mentioned as well by Manning and Schütze [140], whom distinguish between co-occurrence (or association) and collocation:

collocation refers to 'grammatically bound elements that occur in a particular order', but *co-occurrence* and *association* refer to 'the more general phenomenon of words that are likely to be used in the same context'. Mainly, when it refers to synonyms order does not matter, so they co-occur. At this point we are more interested in the idea of terms that are *collocated*, and Pointwise Mutual Information (PMI) has been utilised extensible in the analysis of collocation. Let us introduce firstly some basic concepts, most of which belong to the category of Vector Space Models (VSMs) of Semantics [216, 231].

The *Euclidean Distance* between vector x and y of dimension n is defined as

$$\sqrt{\sum_{i=1}^{n} |x_i - y_i|^2} \tag{10.4}$$

In turn, the Cosine Distance between vectors u and v of n dimensions is defined as

$$\text{Cosine Distance}(u, v) = 1 - \text{Cosine Similarity}(u, v) \tag{10.5}$$

where,

$$\text{Cosine Similarity} = cos(\theta) = \frac{A \cdot B}{\| A \| \| B \|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{[\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}} \tag{10.6}$$

where, the operator '$\cdot$' stands for the dot product or scalar product, and $\| v \|$ corresponds to the magnitude of the vector v.

The concept of *neighbourhood* between two words is of significant importance as well. Let us assume we do have a matrix m, *word* x *word*, where the *count* in cell $m[i, j]$ is the number of times that word i and word j occurred together in the same text. Then the function *neighbour* allows to understand what are the closest word to a given word w_k, according to the concept of Cosine distance defined above. Some parts of the information in this sub-section have been obtained from (`http://web.stanford.edu/class/cs224u/`), CS 224U (LING 188/288) Natural Language Understanding, Spring 2014 at Stanford University.

According to Turney and Littman [213, 215] it is possible to infer semantic orientation from semantic association. As such, they defined seven positive words [*good*, *nice*, *excellent*, *positive*, $fortunate$, *correct*, *superior*] and seven negative words [*bad*, *nasty*, *poor*, *negative*, $unfortunate$, *wrong*, $inferior$] that are used as a *paradigm* for positive and negative semantic orientation. "The semantic orientation of a given word is calculated from the strength of its association with the seven positive words, minus the strength of its association with the seven negative words" [213]. The concept of using PMI to calculate the strength of the semantic association between words stems from Church and Hanks [60]. The Pointwise Mutual Information (PMI) between two words, $word_1$ and $word_2$, is defined by Church and Hanks [59, 60] as:

$$PMI(word_1, word_2) = \log_2 \left(\frac{p(word_1) \;\&\; p(word_2)}{p(word_1)\;\; p(word_2)} \right) \tag{10.7}$$

where $p(word_1)$ & $p(word_2)$ corresponds to the probability that $word_1$ and $word_2$ co-occur. The product $p(word_1)$ $p(word_2)$ compensates for the case that the words are statistically independent. The ratio between both components is "a measure of degree of statistical dependence between the words. The log of the ratio is the amount of information that we acquire about the presence of one word when we observe the other". Then, it follows that the semantic orientation of a word, *word*, is calculated by SO-PMI-IR as follows [59, 60]:

$$\text{SO-PMI-IR}(word) \;=\; \text{PMI}(word, PositiveParadigms) - \text{PMI}(word, NegativeParadigms) \tag{10.8}$$

where,

PositiveParadigms = { good, nice, excellent, positive, fortunate, correct, superior }, and

NegativeParadigms = { bad, nasty, poor, negative, unfortunate, wrong, inferior }

D. Terence Langendoen [125] says in his review of some of the work of the famous English linguist J. R Firth [84]: "His notion of meaning by collocation, however, deserves careful attention. Firth viewed this notion as an attempt to formalize within semantic theory Wittgenstein's thesis that 'the meaning of words lies in their use' ." Hence, we see that *collocation* and *semantic* enjoy a very peculiar link. Basically, "words that occur in similar contexts tend to have similar meanings" [216]. We will attempt to exploit that link by using *PMI* to determine how close semantically a word x is to another word y. However, there are some issues that must be addressed first. According to Turney and Pantel [216] "computers understand very little of the meaning of human language". This situation translates into challenges for humans to give orders to computers and limits the capabilities of computers to analyse, process and understand text. As per [216], the so-called Vector Space Models of Semantics (VSMs) are beginning to address the limitations we have just mentioned (statement circumscribed to the time period the article was published: 2010). The main idea behind Vector Space Models, as developed by Salton et al. [190, 191], is that one can represent each document in a collection as a point in space (a vector in vector space). "Points that are close together in this space are semantically similar and points that are far apart are semantically distant. The user's query is represented as a point in the same space as the documents" [216]. The success of VSM in information retrieval has encouraged other researchers to explore other uses of VSM in NLP. As already mentioned, it has been reported [216] that VSM performs very well on processes orientated to measuring the similarity of meaning between words -phrases and documents-. Turney and Pantel [216] claim that the leading algorithms for measuring semantic relatedness use VSMs [166].

There are a number of tools in the VSM arsenal, and the PMI is one that is very efficient at finding word similarity. "Each word is represented by a feature vector. Each feature corresponds to a context in which the words occurs" [166]. Pantel and Lin continue: "For example, "sip _ _ _ _" is a verb-object context. The word *wine* occurred in this context, the context is a feature of *wine*. The value of a feature is the pointwise mutual information (PMI) between the feature and the word. Let c be a context and $F_c(w)$ be the frequency count of a word w occurring in context c. The pointwise mutual information, $mi_{w,c}$ between c and w is defined as":

$$mi_{w,c} = \frac{\frac{F_c(w)}{N}}{\frac{\sum_i F_i(w)}{N} \times \frac{\sum_j F_c(j)}{N}} \tag{10.9}$$

where $N = \sum_i \sum_j F_i(j)$ is the total frequency counts of all words and their contexts. As per the authors, a well-known problem with mutual information is that it is biased towards infrequent words or features. As a consequence, $mi_{w,c}$ is multiplied with a *discounting factor*:

$$\frac{F_c(w)}{F_c(w)+1} \times \frac{\min\left(\sum_i F_i(w), \sum_j F_c(j)\right)}{\min\left(\sum_i F_i(w), \sum_j F_c(j)\right)+1} \tag{10.10}$$

Pantel and Lin [166] close by stating "We compute the similarity between two words w_i and w_j using the *cosine coefficient* of their mutual information vectors":

$$sim(w_i, w_j) = \frac{\sum_c mi_{w_i c} \times mi_{w_j c}}{\sqrt{\sum_c mi^2_{w_i c} \times \sum_c mi^2_{w_j c}}} \tag{10.11}$$

We have shown the basis for using PMI and its mathematical foundations. In our research work we will be using the PMI *discounting factor* version as mentioned above to obtain relationships of attraction or gravity for the words that are part of our Opinion Lexicon. This 'negative gravity' and 'positive gravity' as we have decided to call them, represent the average distance between a given word and the *PositiveParadigms* seeds and the *NegativeParadigms* seeds, respectively.

At Stanford University, Chris Potts [178] generated some Python code for VSM and prepared some clean data sets based on data available at IMDB (`http://www.imdb.com/`), the so-called *Internet Movie Database*. We see that as per [178],

there is available a frequency matrix with a structure as shown below in Table 10.3. It is a $Word \times Word\ Matrix$, where the **count** in cell $m[i, j]$ corresponds to the number of times that Word i and Word j occurred together in the same text (called $c[i, j]$ in Table 10.3). The file prepared by Prof. Potts, named 'imdb-wordword.csv' contains a (2,998 x 2,998) (Word x Word) matrix, representing a Frequency Matrix generated from movie reviews available in IMDB. For convenience, we will use this data and the VSM Python code provided by Prof. Potts, for our experiments and tests and for extracting information to augment the richness of our Opinion Lexicon. The process that we followed to generate these

Count	$Word_1$	$Word_2$	$Word_k$	$Word_n$
$Word_1$	c_{11}	c_{12}	c_{1k}	c_{1n}
$Word_2$	c_{21}	c_{22}	c_{2k}	c_{2n}
...	...	...	...	...
$Word_n$	c_{n1}	c_{n2}	c_{kn}	c_{nn}

Table 10.3. VSM Word x Word Matrix

aforementioned 'positive gravity' and 'negative gravity' coefficients is as follows:

1. Obtain the matrix m containing the Word x Word structure detailing frequency, as described above.
2. Generate the PPMI (Positive PMI with discounting adjustment) matrix p, as described in [211].
3. For every word in matrix p, calculate the *cosine distance* with respect to the set of *positive seeds* and the set of *negative seeds*. For every word in p we will generate distance values: one telling us 'how close' a given word w_k is from being *semantically positive* (Maxdist) and another telling us 'how far' it is from that (Mindist) -or in other words, how close w_k is to the cluster of negative seeds-.
4. Apply the concept of Semantic Orientation (CSOR) described by Potts in [179] and calculate the SO of the words in matrix p.
5. From all these scores described above, Maxdist, Mindist and CSOR, incorporate them into our opinion lexicon (for those words that are actually included already in the lexicon as sentiment-carrying words).

Note: in Section 14.1.1 we describe the final version of our sentiment lexicon and its attributes. However, attributes *Maxdist*, *Mindist* and *CSOR* are not part of it. After a number of experiments, we noticed that we did not have enough data to populate these aforementioned three variables. As such, they became very hard to use efficiently in our proposed model. Hence, we decided to *reject* these attributes from the lexicon and continue using the sentiment lexicon described in Section 14.1.1 on this report. For completeness, we have decided to include the Vector Space Models (VSM) of Semantics & PMI in this report, as they were part of our research and they looked promising. Nevertheless, as mentioned already the lack of data made these techniques unusable in our proposed model.

10.3 Latent Semantic Analysis (LSA) Method

Turney & Littman [214] addressed as well the inference of semantic orientation from association. According to Turney, as mentioned before, a word's orientation is associated to its neighbour words' orientation. As such, the Semantic Orientation of a given word, by association, **SO-A**, is proposed as follows:

$$SO-A(word) = \sum_{pword \in Pwords} A(word, pword) - \sum_{nword \in Nwords} A(word, nword) \tag{10.12}$$

where,

- $Pwords$: a set of words with positive semantic orientation.
- $Nwords$: a set of words with negative semantic orientation.
- $A(w_1, w_2)$: a measure of association between w_1 and w_2.
    - Maps to a real number.

- Positive/Negative - Presence/Absence.

The following seven positive and seven negative words are defined as "paradigms of positive and negative semantic orientation" [214]:

- *Positive set*: good, nice, excellent, *positive*, fortunate, correct and superior.
- *Negative set*: bad, nasty, poor, *negative*, unfortunate, wrong and inferior.

For Turney & Littman, the above paradigm words represent semantic orientation rather than training the LSA algorithm. According to the authors, SO-LSA applies Latent Semantic Analysis (LSA) in order to compute the intensity or strength of the semantic link that exists between two words. LSA utilises an algorithm called Singular Value Decomposition [214] with the objective of analysing the statistical relationships among all words in a given corpus. The text being analysed is initially used to build a matrix, in which the rows represent words in the corpus, whilst the columns correspond to words in the corresponding chunk of text. For more information on the mechanics of this method, refer to [216, 231].

10.4 Word-frequency Lists / Dictionaries

Technically speaking, a Word-frequency Dictionary is not an UML method. However, it is a strategy that has been used in a number of situations [26, 135, 164, 165, 173, 177, 231], and we make use of it in our proposed model (see Chapter 14). In our case, we opted for creating a word-frequency dictionary with the following structure and mnemonics (W=word, NO= Number of occurrences in corpus, SN = pointer to sentence number, PR = Polarity, either Positive or Negative):

[W NO $\{(SN_1, PR) \ldots (SN_n, PR)\}$], where n = number of words that are part of the dictionary.

Notice that the third element in the dictionary is a list of pairs, where the first value corresponds to a pointer to a sentence where the word in question is present, and the second one provides the polarity of the sentence to which the first value is pointing. The application of this strategy in assisting in polarity identification, does require two steps. In this first step, the whole dictionary is created. In the second one, the dictionary is used to look for those cases where the following condition is satisfied: for the sentence being processed, not even one of the participating words exist in the lexicon. As such, we will attempt to determine the polarity of the words by computing how many times they have been spotted in sentences that have yielded previously a specific polarity (either positive or negative).

If **Count** (Positive Occurrences) > **Count** (Negative Occurrences) then the Sentence being Processed will get assigned a *Positive* polarity
If **Count** (Positive Occurrences) < **Count** (Negative Occurrences) then the Sentence being Processed will get assigned a *Negative* polarity
Otherwise, the sentence will be assigned an *Objective* polarity.

For a corpus of n words, this algorithm is computationally expensive as it will have a Θ^2 order. See sub-section 14.1.4.1.1 for more information on when and how this technique is utilised in our proposed solution.

10.5 Other Methods

Among other options, Rothfels & Tibshirani [185] proposed an Unsupervised Method based on a revised version of PMI and other components. More recently Blein et al. [28] proposed the Latent Dirichlet Allocation (LDA) algorithm (a generative model that allows sets of observations to be explained by unobserved groups that explain why some parts of data are similar). Reed [181] provides a good tutorial on LDA. Usha and Devi [218] present a method based on Gibbs Sample procedure (a Markov chain Monte Carlo algorithm for obtaining a sequence of observations which are approximated from a specified multivariate probability distribution). In [128] a Weakly Supervised joint sentiment-topic (JST) detection method is presented, which according to the authors is a novel probabilistic modeling framework based on latent Dirichlet allocation (LDA) [28] and that is supposed to be capable of detecting simultaneously sentiment and

topic out of a given text (as the aforementioned method is classified as *weakly-supervised* we have arbitrarily decided to placed it under Unsupervised Methods).

10.6 Chapter Summary

In this chapter we have discussed unsupervised machine learning techniques. As we have covered as well supervised machine methods in the previous chapter, this concludes the content related to machine learning. In the next chapter we will aboard the topic of the fundamental concepts of emotions.

Chapter 11

The Concept of Emotions

> "I wish it need not have happened in my time", said Frodo. "So do I", said Gandalf, "and so do all who live to see such times. But that is not for them to decide. All we have to decide is what to do with the time that is given us."
>
> J.R.R. Tolkien, The Fellowship of the Ring.

In this chapter, we are looking at sentiments from the perspective of the theory of emotions, which is rooted in neurological and psychological foundations. We have addressed this topic as we were interested in understanding how these aspects could impact, if at all, the classification of sentiments.

11.1 The Ortony-Clore-Collins (OCC) Model

According to Ortony et al. [161] the cognitive structure of emotions must have some specific characteristics: "We share Abelson's (1983) view that an analysis of emotion must go beyond differentiating positive from negative emotions to give a systematic account of the qualitative differences among individual emotions such as fear, envy, anger, pride, relief and admiration". In [161], the authors attempt to establish the cognitive antecedents of emotion and strive to produce systematic explanations. In time, the Ortony-Clore-Collins model (OCC) has become a reference and baseline for most of the work conducted in the field. According to Schnoebelen [193], Ortony et al. start out with a notion of *eliciting conditions*, which to be effective "the experiencing individual must encode the relevant situation in a particular way.". Schnoebelen continues saying that "the OCC model propose an arousal-producing mechanism that simultaneously registers valence". The main generalisations that Ortony et al. find in the past literature are [193]:

- Emotion involves arousal and appraisal.
- Any dimensional characterization of emotion is likely to include at least activation and valence.
- There are basically two sides of a coin; activation = arousal and valence = appraisal.
- These dimensions are, for OCC, uninformative/unsurprising.

Emotions are "valenced reactions to events, agents, or objects, with their particular nature being determined by the way in which the eliciting situation is construed" [161, pp. 13]. According to Schnoebelen [193], this is a refinement of an earlier notion that "emotions are internal, mental states that vary in intensity and that are focused predominately on affect. By **affect** we simply mean evaluative reactions to situations as good or bad" [161, 190-191]. An *Emotion type* is "a distinct kind of emotion that can be realized in a variety of recognizably related forms" [193].

Emotion types are divided by whether they focus on events, agents, or objects [193]. "**Events** are simply people's construals about things that happen, considered independently of any beliefs they may have about actual or possible cause. **Objects** are objects viewed qua-objects. This leaves us with **agents**, which are things considered in light of their actual or

presumed instrumentality or agency in causing or contributing to events. Agents can be non-human animate beings, inanimate objects or abstractions, such as institutions, and even situations, provided they are construed as causally efficacious in the particular context" [161, pp.18]. Figure 11.1 reproduces the *global structure of emotion types* as per Ortony et al.

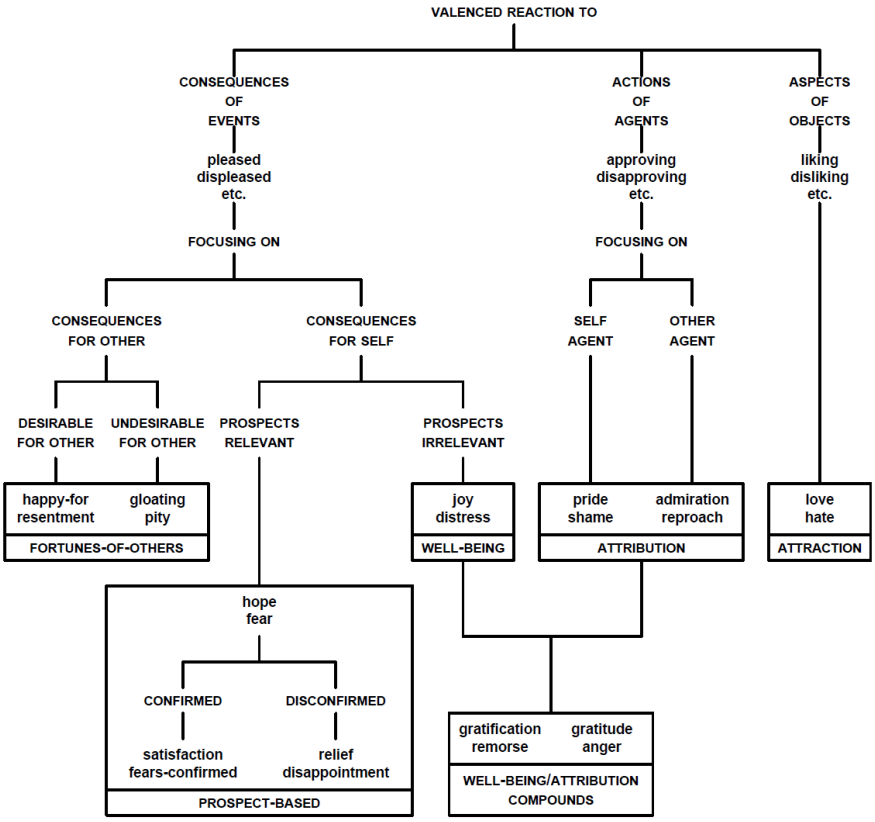


Fig. 11.1. Original Structure of Emotions of the OCC model ([161, pp. 19], re-illustrated from [201])

For Ortony et al. there are global factors that determine emotional intensity [193]:

- Sense of reality (how much do you believe the emotion-inducing situation is real?)
- Proximity (how close in psychological space do you feel to the situation?)
- Unexpectedness (how surprised are you by the situation?)
- Arousal (how much are you aroused prior to the situation?)

The structure proposed by Ortony et al. is the so-called OCC Model, as depicted in Fig. 11.1. There are three main branches stemming from a tree and they represent the three different ways in which an individual can react to the world. The representation is logical and does not follow any temporal approach. Each one of the aforementioned branches are linked to broad classes of affective reactions. The intensity of the emotions will dictate whether or not they will be interpreted as such, hence the importance of knowing which are the factors affecting the intensity of such feelings. These aforementioned reactions can have a valence (representing *intensity*) associated to them [161, 19-20].

11.2 OCC Revisited

Despite the fact that the OOC Model is considered to be a very solid one, some authors considered that the model includes some minor ambiguities that should be removed if one were to attempt to use the OCC Model for computational purposes. As such, Steunebrink et al. [202] have proposed the so-called OOC Model Revisited, to identify and clarify several of the aforementioned ambiguities. In addition, they introduce a new inheritance-based view of the logical structure of emotions

of the OCC model proposed by Ortony et al. [161], which is depicted in Fig. 11.2.

When commenting on the main differences between the Revisited OCC and the original one, the authors claim the

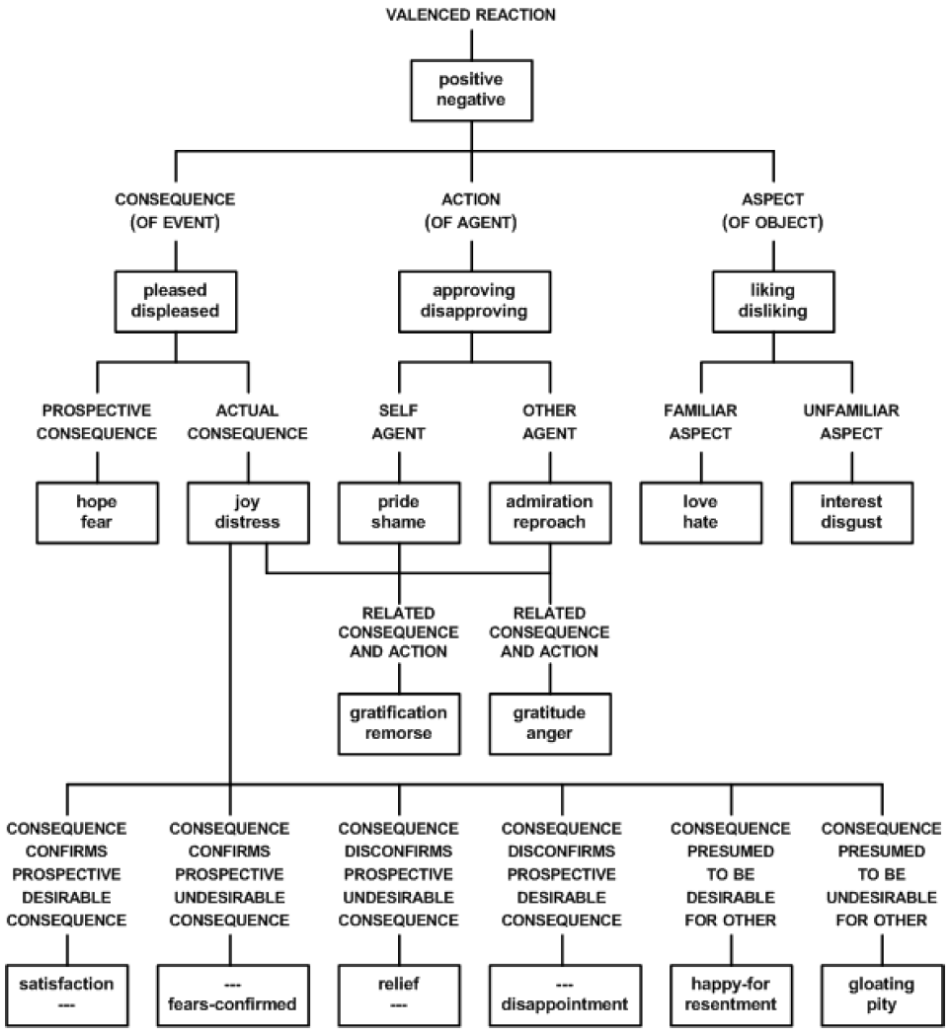


Fig. 11.2. A disambiguated, inheritance-based hierarchy of emotions of the OCC model ([202, pp. 7]).

following (taken from [202]):

1. The Revisited OCC Model “represents the inheritance structure explicitly, because it has labels at every point in the hierarchy, and the conditions of every child node are a superset of those of its parent node(s)”.
2. “Along each connection are written (in small caps) the additional condition(s) that make an emotion type a specialization of its parent type(s). Ambiguous terms are avoided; for example, we have omitted or replaced the phrases ‘focusing on’ and ‘prospects (ir)relevant’”.
3. “Satisfaction, fears-confirmed, relief, and disappointment have been moved from under hope/fear to become specializations of joy/distress. This is because we regard a confirmation or disconfirmation to be an actual consequence (of an event)”.
4. “Happy-for, resentment, gloating, and pity have been moved from the left of” the original OCC model “to the bottom right of” the Revisited OCC model.
5. As “love/hate does not seem to extend its parent type”, “we have added interest/digust as an additional (i.e., besides love/hate) specialization of liking/disliking, based on the familiarity with the object in question”.

6. "One thing that we find particularly attractive about", the Revisited OCC Model, "is that type specifications now follow immediately from the diagram. Descriptions can easily be formed by following any link from child to parent node, inserting the text on the link.". The resulting type specifications are displayed in Table 11.1.

positive	and *negative* are valenced reactions (to "something")
pleased	is being *positive* about a consequence (of an event)
displeased	is being *negative* about a consequence (of an event)
hope	is being *pleased* about a prospective consequence (of an event)
fear	is being *displeased* about a prospective consequence (of an event)
joy	is being *pleased* about an actual consequence (of an event)
distress	is being *displeased* about an actual consequence (of an event)
satisfaction	is *joy* about the confirmation of a prospective desirable consequence
fears-confirmed	is *distress* about the confirmation of a prospective undesirable consequence
relief	is *joy* about the disconfirmation of a prospective undesirable consequence
disappointment	is *distress* about the disconfirmation of a prospective desirable consequence
happy-for	is *joy* about a consequence (of an event) presumed to be desirable for someone else
resentment	is *distress* about a consequence (of an event) presumed to be desirable for someone else
gloating	is *joy* about a consequence (of an event) presumed to be undesirable for someone else
pity	is *distress* about a consequence (of an event) presumed to be undesirable for someone else
approving	is being *positive* about an action (of an agent)
disapproving	is being *negative* about an action (of an agent)
pride	is *approving* of one's own action
shame	is *disapproving* of one's own action
admiration	is *approving* of someone else's action
reproach	is *disapproving* of someone else's action
gratification	is *pride* about an action and *joy* about a related consequence
remorse	is *shame* about an action and *distress* about a related consequence
gratitude	is *admiration* about an action and *joy* about a related consequence
anger	is *reproach* about an action and *distress* about a related consequence
liking	is being *positive* about an aspect (of an object)
disliking	is being *negative* about an aspect (of an object)
love	is *liking* a familiar aspect (of an object)
hate	is *disliking* a familiar aspect (of an object)
interest	is *liking* an unfamiliar aspect (of an object)
disgust	is *disliking* an unfamiliar aspect (of an object)

Table 11.1. Emotion type specifications corresponding to Fig. 11.2 (as reproduced from [202, pp. 8]).

This Revisited OCC Model seems to be a better fit for those attempting to produce any sort of computational model of emotions, because emotions are formalised "in a way which is both truthful to the OCC model and useful for Artificial Intelligence" [202].

11.3 Other models of emotion

Personally speaking, we believe that the best way to understand human emotions is reading the works of Homer, Dante and Shakespeare. However, we realise that this style of learning may lack the level of formality required in research initiatives. It is commonly believed that one of the first documents dealing with the origin and evolution of emotions and related topics can be tracked down to Charles Darwin with his treaty on primary emotions [69]. James [105, 106] and more recently Plutchik [174] and Ekman [76] produced dramatic contributions to this field of study. Let us look at some of the contributions provided by some of these authors.

11.3.1 Darwin

One of the main contributions of Charles Darwin to this topic was his idea of looking at emotions as an evolutionary aspect. In his work published in 1872 [69], Darwin introduced the concept of *basic emotions*. According to Charles

Darwin, emotions are biologically determined [69] and they establish a common link among beings. All in all, his ideas germinated in a number of researchers whom carried on with the work generating many other theories.

11.3.2 Plutchik

In [29], Blewitt discusses the main ideas proposed by Plutchik [174, 175]. This researcher is recognised as having introduced the so-called eight *basic emotions* (joy versus sadness; anger versus fear; trust versus disgust; and surprise versus anticipation) and *advanced emotions* (those that result when a given subject experiences in parallel a number of emotions). Plutchik argued that the eight basic emotions are bipolar and that emotions can be expressed at different intensities (the latter concept is interesting to us as it ties to the idea of graduality). Plutchik's psycho-evolutionary theory of basic emotions includes ten postulates [175]:

1. The concept of emotion is applicable to all evolutionary levels and applies to all animals including humans.
2. Emotions have an evolutionary history and have evolved various forms of expression in different species.
3. Emotions served an adaptive role in helping organisms deal with key survival issues posed by the environment.
4. Despite different forms of expression of emotions in different species, there are certain common elements, or prototype patterns, that can be identified.
5. There is a small number of basic, primary, or prototype emotions.
6. All other emotions are mixed or derivative states; that is, they occur as combinations, mixtures, or compounds of the primary emotions.
7. Primary emotions are hypothetical constructs or idealized states whose properties and characteristics can only be inferred from various kinds of evidence.
8. Primary emotions can be conceptualized in terms of pairs of polar opposites.
9. All emotions vary in their degree of similarity to one another.
10. Each emotion can exist in varying degrees of intensity or levels of arousal.

11.3.3 Ekman

Paul Ekman, was an American psychologist who is considered to be a pioneer in the study of emotions and their relation to facial expressions [76]. Below we present the so-called Basic Emotions that Ekman proposes in his theory (see Table. 11.2). Ekman affirms that the number of basic emotions is limited. However, he claims as well that there are

Name	**Definition**
Anger	A strong feeling of annoyance, displeasure, or hostility
Disgust	A feeling of revulsion or strong disapproval aroused by something unpleasant or offensive
Fear	An unpleasant emotion caused by the threat of danger, pain, or harm
Happiness	Feeling or showing pleasure or contentment
Sadness	Feeling or showing sorrow; unhappy
Surprise	A feeling of mild astonishment or shock caused by something unexpected

Table 11.2. Basic Emotions - Ekman's Theory [29, 76]

'non-basic' emotions that happen to be combinations of the aforementioned basic emotions.

11.4 Non-psychological models

There are some models of emotions that are not based on psychological backgrounds. However, for completeness we will mention two other lines of thinking about emotions.

11.4.1 Neurobiological Background

Neurologist António Damásio [68], provides an extraordinary account of his challenge with respect to one of the original ideas of Descartes (the human mind as separate from bodily processes). In [68], Damásio, among many other complex situations, argues that there are *primary and secondary emotions*, and provides a classification as follows (as *quoted* from [24]):

- Primary emotions (fear, anger, joy, etc.):
  - fast, hardwired stimulus response patterns
  - trigger fight or flight behaviors
  - ontogenetically earlier types of emotion
- Secondary emotions (hope, shame, etc.)
  - lead to cognitively elaborated, deliberative behaviors
  - are based on memories and expectations
  - 'social emotions' developed during infancy
  - 'utilize the machinery of primary emotions'

11.4.2 Social/Interpersonal Background

According to Parkinson et al. [167] "emotions are not necessarily defined by the quality of the associated feeling state but may instead derive their identity from the interpersonal dynamics that provide the context for their subjective aspects". The authors highlight as being of particular interest the so-called 'sociomoral emotions' [24, 167].

- Embarrassment: 'an interruption of the orderly performance of social action'.
- 'Shame': a person's 'failure to live up to central standards of conduct [in the eye of others]'.
- Guilt: 'blameworthy action is the key elicitor'.

For the curious reader, please refer to [167] for complete details.

11.5 Chapter Summary

In this chapter we have addressed some concepts related to the theory of emotions. We have made some decisions related to the potential incorporation of these aspects into our proposed classification model (see Chapter 17), and the concepts described in this chapter, were important in the decision-making process. In the next chapter we will cover an important research field that has proven to be instrumental in our research and associated results: *Fuzzy Reasoning*.

Chapter 12

Fuzzy Reasoning in Sentiment Analysis

"Vivimos en un mundo esencialmente apócrifo, en un cosmos o poema de nuestro pensar, ordenado o construido todo él sobre supuestos indemostrables, postulados de nuestra razón, que llaman principios de la lógica, los cuales, reducidos al principio de identidad que los resume y reasume a todos, constituyen un solo y magnífico supuesto: el que afirma que todas las cosas, por el mero hecho de ser pensadas, permanecen inmutables, ancladas, por decirlo así, en el río de Heráclito. Lo apócrifo de nuestro mundo se prueba por la existencia de la lógica, por la necesidad de poner el pensamiento de acuerdo consigo mismo, de forzarlo en cierto modo, a que sólo vea lo supuesto o puesto por él, con exclusión de todo lo demás. Y el hecho -digámoslo de pasada- de que nuestro mundo esté todo él cimentado sobre un supuesto que pudiera ser falso, es algo terrible, o consolador. Según se mire. Pero de esto hablaremos otro día."

Antonio Machado (Juan de Mairena, 1936).

"When we are not sure, we are alive."

Graham Greene.

In this chapter the fundamentals of the application of fuzzy sets to the SA problem are discussed. We start by briefly addressing some fuzzy sets and fuzzy logic basic concepts, and we close the chapter presenting some real cases of the utilisation of fuzzy reasoning techniques in the SA field.

According to Zadeh [261, 264, 265] fuzzy sets are incredibly helpful when information is incomplete or imprecise and ambiguity is present. But, what could be at the same time imprecise and ambiguous and in another occasions, very precise? There is no doubt that Natural Languages fit this description. As such, we believe that there is certainly lots of merits in considering fuzzy sets/logic as a tool that could perform well in NLP-related problems.

12.1 Fuzzy Sets

In this section we will briefly refresh the mind of the reader about the main ideas involving fuzzy sets and fuzzy logic. Fuzzy Sets were introduced by Lotfi A. Zadeh in 1965 [261]. As covered in [119], *if X* is the universal set defined in a specific problem, with elements denoted generally by x, then a *fuzzy set* **A** in X is a set of ordered pairs:

$$A = \{(x, \mu_A(x)) \,|\, x \in X\} \tag{12.1}$$

where $\mu_A : X \rightarrow [0, 1]$ is called the *membership function* of A and $\mu_A(x)$ represents the degree of membership of the element x in A. Notice that if the range of the membership function is enforced to be either 0 or 1, then the fuzzy set becomes a regular (crisp) set. Now let us take a look at the basic operations with fuzzy sets which extend classical set theory consistently.

1. **Intersection (min operator)**:

   $$\mu_{A \cap B}(x) = min\{\mu_A(x), \mu_B(x)\}$$

   that is equivalent in classical sets to

   $$x \in A \cap B \Leftrightarrow x \in A \wedge x \in B$$

2. **Union (max operator)**:

   $$\mu_{A \cup B}(x) = max\{\mu_A(x), \mu_B(x)\}$$

   that is equivalent in classical sets to

   $$x \in A \cup B \Leftrightarrow x \in A \vee x \in B$$

3. **Complement**:

   $$\mu_{\bar{A}} = 1 - \mu_A(x)$$

t-norms and *t-conorms* -'t' for triangular- were introduced to enable generalisation from boolean to multi-valued logic. *t-norms* define a general class of *intersection* operators for fuzzy sets, whilst *t-conorms* define a general class of operator for performing the *union* of fuzzy sets.

A **t-norm** is a two valued function

$$T\,[0, 1]\,x\,[0, 1] \rightarrow [0, 1]$$

which satisfies the following conditions:

1. $T(x, 1) = x \;\forall x$ (boundary)
2. $T(x, y) = T(y, x) \;\forall x, y$ (commutativity)
3. $T(x, y) \leq T(x, z)\; if\; y \leq z$ (monotonicity)
4. $T(x, T(y, z)) = T(T(x, y), z)$ (associativity)

The most commonly used t-norm for fuzzy intersection is the *minimum (min)* operator.

A **t-conorm** is a two valued function

$$S\,[0, 1]\,x\,[0, 1] \rightarrow [0, 1]$$

which satisfies the following conditions:

1. $S(x, 0) = x \;\forall x$ (boundary)
2. $S(x, y) = S(y, x) \;\forall x, y$ (commutativity)
3. $S(x, y) \leq S(x, z)\; if\; y \leq z$ (monotonicity)
4. $S(x, S(y, z)) = S(S(x, y), z)$ (associativity)

The most commonly used t-conorm for fuzzy union is the *maximum (max)* operator.

The work of De Cock et al. [63] and De Cock & Kerre [62] is very illustrative about modelling linguistics expressions using fuzzy relations and introducing 'fuzzy modifiers' for *adverbs* applying fuzzy relations.

12.2 Fuzzy Logic

Fuzzy logic is a form of many-valued logic. As introduced by Lotfi A. Zadeh "it deals with reasoning that is approximate rather than fixed and exact" [261]. As fuzzy logic is based on fuzzy sets, its variables may have a truth value that ranges in degree between 0 and 1 in $\mathbb{R}$, as opposed to those of classical logic based on binary sets, admitting exclusively either a value of 0 or 1. Another great advantage of fuzzy logic is the enabling of the use of *linguistics variables*. In real life situations, experts have to deal with vague, ambiguous or imprecise information or have to express their opinions on qualitative aspects. According to Zadeh, "Variables whose values are not numbers but words or sentences in a natural or artificial language. The motivation for the use of words or sentences rather than numbers is that linguistic characterizations are, in general, less specific than numerical ones".

"As an extension of the case of multi-valued logic, valuations ($\mu : V_o \rightarrow W$) of propositional variables (V_o) into a set of membership degrees W can be thought of as membership functions mapping predicates into fuzzy sets (or more formally, into an ordered set of fuzzy pairs, called a fuzzy relation). With these valuations, many-valued logic can be extended to allow for fuzzy premises from which graded conclusions may be drawn" [90]. This extension is sometimes called 'fuzzy logic in the narrow sense' as opposed to 'fuzzy logic in the wider sense', which originated in the engineering fields of automated control and knowledge engineering, and which encompasses many topics involving fuzzy sets and 'approximated reasoning'. [263]. For more details on fuzzy logic, please refer to [38, 90, 118, 119, 184, 261, 264, 269].

12.3 Fuzzy Sets applied to the SA Problem

In this section we will proceed in two steps. Firstly, we will describe the elements that we believe can be used to apply *fuzzy sets* to the SA problem. Secondly, we will present some of the documented cases of actual applications of fuzzy sets to the SA problem.

12.3.1 Fuzzy Sets in SA

It is well-known that *linguistic variables* and linguistic characterisations are less precise that numerical ones. It is sort of intuitive to think of fuzzy sets as good tools to model natural languages. But how? Is it the fuzzification of the dictionary? Is it the introduction of fuzziness in the polarity range in the sentiment lexicon? Is it a different approach, like using fuzzy sets for subjectivity identification or only for polarity clarification? Or both, perhaps?

In the sub-sections that follow we will cover more details on the better known cases on the use of fuzzy sets in SA. However, we would like to establish early enough our leaning. In our mind the most interesting use of fuzzy sets in the SA problem, would be linked to the introduction of fuzziness at the heart of the linguistic/lexicon-based approach. That would imply using fuzzy sets in the potential creation of a sentiment lexicon capable of managing graduality in the polarity attribute (i.e. $\{excellent, verygood, good\}$), all being possible states of a positive opinion, but certainly not all three of them expressing the same opinion strength. In the case of identifying subjectivity, we tend to believe that there is merit on looking at both options: (a) using fuzzy sets, and (b) utilising NLP linguistic techniques (based most likely on *semantic rules* or unsupervised learning). This way we would be in the presence of a *hybrid* approach where fuzzy sets and linguistic methods would be applied. We could even extrapolate that it would be reasonable to use linguistic approaches to predict polarity and to apply fuzzy sets to 'validate' the previous assessment, and further on, to establish the strength or intensity of the polarity case, once the latter has been identified appropriately. We will elaborate more on this approach in Part IV.

12.3.2 Fuzzy Sets Cases

There are a few approaches to the utilisation of fuzzy sets in the SA problem that has been documented. We will cover in the next paragraph some of them, but cannot guarantee that we will cover all of them. We will detail those that we consider to be of greater significance because of the approach they have followed or because they claim to have found results that are comparable -or better- than those exhibited in research based on Supervised Machine Learning (SML).

12.3.2.1 Case 1: Affect Analysis using Fuzzy Semantic Typing

Subasic and Huettner proposed in [204, 205] an approach to affect analysis using what they called fuzzy semantic typing. Most of the contents of this sub-section has been extracted from the work they published in 2001 [205]. The authors seem to have given their deepest thoughts to the problem of applying fuzzy sets to identify emotions in an automated fashion. Their proposal is based on a fusion of traditional NLP tools and fuzzy logic techniques aimed to be used to decipher affect content in text. As per Subasic and Huettner, the linguistic resources used by the fuzzy typing system are: the affect-lexicon, the so-called fuzzy-thesaurus and what they call affect-category groups.

- Affect Lexicon is defined as a collection of entries for terms that mean affects. In addition, every entry for an affect term, includes additional key data like the parts-of-speech that word belong to, its affect categories, its centralities and its intensities. Entries in the affect lexicon follows this form:

  $\langle lexical\_entry \rangle \langle part\_of\_speech\_tag \rangle \langle affect\_category \rangle \langle centrality \rangle \langle intensity \rangle$

  as in the example (‘arrogance’ *sn superiority* 0.7 0.9). In the *affect lexicon* just mentioned, we must clarify the meaning of a few items.
  Lexical_entry is a single entry for terms that carry emotional connotations or directly imply an emotion/affect.
  Part_of_Speech_tag: understanding what part-of-speech a given term belongs to is critical, otherwise it would be very hard to resolve ambiguity issues. For instance, the word ‘alert’ could mean ‘intelligence’ (when it is an *adjective*) or it could imply a ‘warning’ (when it is a *verb*). Subasic and Huettner claim that a word’s PoS could affect its *centrality*, its *intensity* or/and its *category* values. The following example provided by the researchers is very illustrative. The word ‘craze’ has the following two entries in their dictionary: (‘craze’ vb insanity 0.8) and (‘craze’ sn insanity 0.5). Then, ‘craze’, as a *verb* belongs to the *affect category* ‘insanity’ with a degree of 0.8; and ‘craze’, as a *noun* (in singular) belong to the same category, but with a degree of 0.5 instead. In this case, Subasic and Huettner claim that this situation is an evidence that “the *verb* ‘craze’ means to *make insane or as if insane* -very central to the *insanity* category!- while the *noun* ‘craze’ means *an exaggerated and often transient enthusiasm*- i.e., it belongs to *insanity* only in a less central way, in a more metaphorical way.” [205].
  Affect_category: the authors offer the disclaimer that several of the categories proposed have wander away from their main affect domain. For example, the words *health, intelligence* and *deprivation* are hardly affects. The same situation present itself with the word *destruction, justice* and *death*. The authors claim that they had to create such categories in order to deal with cases where the meaning of a word cannot be captured using affect categories *and* the same (non-affect) meaning kept appearing in the texts they were trying to analyse. As a consequence, the authors have created some not-strictly-affect categories to deal with these ‘special’ words. Subasic and Huettner claim to have created eighty-three affect categories. They have produced as well an explicit antonym for all categories, with the exception of some words ‘death, irritation, and crime’. See Fig. 12.1 that present a table with all 83 categories as taken from Subasic and Huettner [205].
  Centrality: it ranges from 0 to 1 by increments of 0.1. As such, if a word belongs to more than one affect category it most likely have different centralities, as in: (‘emasculate’ *vb weakness* 0.7), (‘emasculate’ *vb lack* 0.4), and (‘emasculate’ *vb violence* 0.3). But, how is centrality assigned? It seems that all decisions about centrality are passed to the lexicon developer. The authors claim that the questions the lexicon developer should ask herself are: (a) To what extent is *affect* word X related to a given category C? (b) To what extent does *affect* word X co-occur with a given category C? (c) To what extent can *affect* word X be substituted with category C in the text, without changing the semantic? (as quoted from [205]). Centrality computations are performed by obtaining fuzzy membership degrees.

No	Affect Category	Opposite Affect Category
1	absurdity	reasonableness
2	advantage	disadvantage
3	amity	anger
4	anger	amity
5	attraction	repulsion
6	avoidance	desire
7	boredom	excitement
8	clarity	confusion
9	conflict	cooperation
10	confusion	clarity
11	cooperation	conflict
12	courage	fear
13	creation	destruction
14	crime	-
15	death	-
16	deception	honesty
17	desire	avoidance
18	destruction	creation
19	disadvantage	advantage
20	disloyalty	loyalty
21	energy	fatigue
22	excitement	boredom
23	facilitation	prevention
24	failure	success
25	fatigue	energy
26	fear	courage
27	force	persuasion
28	greed	selflessness
29	guilt	innocence
30	happiness	sadness
31	harm	nurturance
32	hate	love
33	health	sickness
34	honesty	deception
35	horror	humor
36	humility	pride
37	humor	horror
38	immorality	morality
39	inferiority	superiority
40	injustice	justice
41	innocence	guilt
42	insanity	sanity
43	insecurity	security
44	insensitivity	sensitivity
45	intelligence	stupidity
46	irresponsibility	responsibility
47	irritation	-
48	justice	injustice
49	lack	surfeit
50	love	hate
51	loyalty	disloyalty
52	morality	immorality
53	nurturance	harm
54	openness	slyness
55	pain	pleasure
56	peace	violence
57	persuasion	force
58	pleasure	pain
59	praise	slander
60	predictability	surprise
61	prevention	facilitation
62	pride	humility
63	promise	warning
64	reasonableness	absurdity
65	repulsion	attraction
66	responsibility	irresponsibility
67	sadness	happiness
68	sanity	insanity
69	security	insecurity
70	selflessness	greed
71	sensitivity	insensitivity
72	sickness	health
73	slander	praise
74	slyness	openness
75	strength	weakness
76	stupidity	intelligence
77	success	failure
78	superiority	inferiority
79	surfeit	lack
80	surprise	predictability
81	violence	peace
82	warning	promise
83	weakness	strength

Fig. 12.1. The Complete List of Affect Categories and Opposite Affect Categories (captured as a figure) as presented in [205]

Intensity: represent the strength of the affect level described by the entry. Intensity degrees range from 0 to 1, with 0.1 increments. In the scores designed by the authors, numbers below 0.4 on the scales of intensity and centrality are those more subjective and notional. The task of assigning values to entries in the lexicon has be assigned to a single linguist professional. As such, the classification will respond to the background, experience and prejudices of one person (the chosen linguist). Even though a rational process is being followed, there is still some subjectivity involved.

- Fuzzy Thesaurus: it is used to establish relationships "between pairs of affect categories, based on the centralities of items assigned to both categories in the lexicon" [205, pp. 486]. The entries in the thesaurus are of the form $\langle affect\_category_1 \rangle$, $\langle affect\_category_2 \rangle$, $\langle relationship\_degree \rangle$, as in 'attraction, love, 0.8' arranged in a matrix. "When the relationship degree is equal to 0, no entry is recorded in the fuzzy thesaurus. When the relationship degree is equal to 1.0 we say that we have discovered *affectual synonyms*" [205], as in 'conflict, violence, 1.0', 'pain, harm, 1.0'. "Non-synonymous pairs having entries in the matrix are *related* to some specified degree". The *fuzzy thesaurus* is produced by the system created by the authors from the affect lexicon. It is created using max-min composition [259].

$$R(AC_i, AC_j) = \bigvee_{A \in \text{AffectLexicon}} \{C_A(AC_i) \wedge C_A(AC_j)\} \tag{12.2}$$

where AC_i, AC_j are affect categories whose relationship degree $R(AC_i, AC_j)$ we want to compute and $C_A(AC_i)$, $C_A(AC_j)$ represent the centralities of affect categories AC_i, AC_j with respect to affect A. The terms $C_A(AC_i)$, $C_A(AC_j)$ are taken directly from the affect lexicon. According to the authors, it is difficult to modify the affect intensity set in a consistent manner, so they leave it to the user to "accommodate the intensities of the added categories for his/her particular purposes". It is important to notice that since the fuzzy thesaurus is generated from

the affect lexicon, any changes in the latter will trigger the need for a re-generation of the fuzzy thesaurus.

- Affect Category Groups: these are automatically generated by clustering the fuzzy thesaurus. Hence, those affect categories that are very similar will get grouped together. As an example the authors mention that 'love', 'attraction', 'happiness', 'desire' and 'pleasure' become one affect category group. Affect category group ACG is a set of affect categories AC_i such that

$$ACG = \{AC_i, i = 1, \cdots, N \mid R(AC_i, AC_j) > R_T, i, j = 1, \cdots, N, i \neq j\} \tag{12.3}$$

where R_T, $0 < R_T \leq 1.0$ is a threshold defined by the user (this ACG is *not* the same similarity relation described by Zadeh in [259]).

Words are assigned specific meta-data from the typing lexicon, as a form of semantic categories and associate degrees, that eventually will produce an outcome for the whole document being analysed. Here is the complete process:

- Affect Sets: it corresponds to the set of affective categories for a given piece of text, inclusive of *intensities* and *centralities*.
- Tagging of Free Text: Subasic and Huettner provides a graphic that depicts the process of generating the affect set, Fig. 12.2, that describes the following steps:

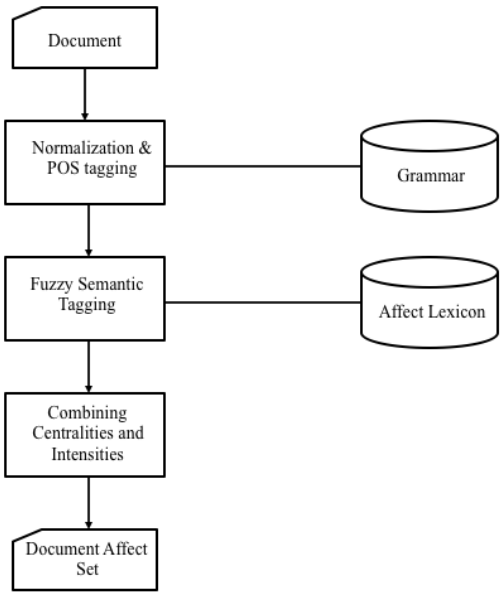


Fig. 12.2. Generation of the affect set for a document: a fuzzy set representing affective content of a document, as re-illustrated from Subasic and Huettner [205].

1. Normalization and tagging: as expected, the document goes through the process of parsing and tokenization, and the resulting tokens are normalised as per the grammar database shown in Fig.12.2. Then the normalised tokens are looked up in the *affect lexicon*. It is important to notice that if a given token has got a number of lexicon entries, all affect categories -with its centralities and intensities- are retrieved. By following this process the initial affect set per document is generated.
2. Combination of Centralities and Intensities and Document Affect Set: here it is the step-by-step process that the authors have created for attempting to reducing the initial affect set (centralities and intensities of recurring categories are combined).
   (a) For each affect category in the tagging set:
      (a.1) "Compute the maximal centrality (fuzzy union) of all centralities attached to that affect category in

the tagged document. The result is the centrality of that category for the document as a whole" [205].

(a.2) "Compute the average intensity of all intensities attached to the affect category in the tagged document. The result is the intensity of that category for the document as a whole" [205].

(b) "Counts of affect categories are combined with intensities using simple averaging to yield the overall intensity score for the document" [205].

As can be appreciated in Fig. 12.3, the algorithm devised by the authors combines "recurring affect categories into a set of unique tags, with centralities and intensities that accurately reflect the overall document content" [205]. It must be noticed the authors' claim, that the number of occurrences of a particular affect category in a given document would affect its *intensity*, but it will not affect its *centrality*. This is a direct consequence of having defined and managed them differently. "*Centrality* indicates the purity of quality represented by an affect category. *Intensity* indicates the strength of that quality. Centrality, as the purity of a quality, depends on the maximal centrality over all instances of that affect category in a particular document. This is to say, the maximal purity of the quality in the document already implies vaguer or more diluted degrees of that quality and, therefore, is appropriate as the combined centrality/purity for that category. The appropriate operation here is thus **fuzzy union**" [205].

Subasic & Huettner decide to compute the intensity attached to an affect category as *the average of all intensities attached to instances of that category*, as "the more times an affect category is present in the document and the higher the intensities of its instances, the higher will be the combined intensity/strength attached to it. We believe this model is closer to how humans perceive intensities of words as they read" [205]. The overall intensity is computed by using a simple average over all affect category instances and their respective intensities

$$I(D) = \sum_{j=1}^{N} \frac{I(ACI_j)}{N} \tag{12.4}$$

where,

$I(D)$ is the overall intensity of a document D;
N is the total number of affect category instances in the document D;
$I(ACI_j)$ is the intensity of an affect category instance ACI_j.

Subasic and Huettner proceed to claim the possibilities that the *fuzzy semantic typing framework* could have, and in order to support their claim, they present the following example. They illustrate their point by showing retrieval of similar phrases that could be extended to retrieval of larger portions of text (i.e. paragraphs or sentences). See Table 12.1 that was produced by running the process we will describe below.

1. The authors found the affect profile of the sentence *fear, anger, grief and pain filled the room*.
2. The authors gathered all phrases from a pre-compiled list of 34 phrases, from four different documents with affect profiles most similar to the provided seed phrase. "As a similarity measure between the affect profile of the seed phrase A and the affect profile of a candidate phrase B", they propose to use (as elaborated in [274]):

$$S(A, B) = \frac{2 \times \| A \cap B \|}{\| A \cup B \|} \tag{12.5}$$

where $\| A \cap B \|$ represents the sum of the intersections (minimum values) of the affect sets' centralities for the respective affect categories and $\| A \cup B \|$ represents the sum of the unions (maximum values) of the affect sets' centralities for the respective affect categories [205].

The authors found sentences, as shown in Table 12.1, where scores for similarity to the seed phrase are ordered by decreasing value. Each phrase is presented with its associated affect centrality and intensity profiles, similarity degrees and average intensities.

Affect Word	POS Tag	Affect Category	Cen t.	Int.
macabre	adj	death	0.5	0.6
macabre	adj	horror	0.9	0.6
comedy	sn	humor	1.0	0.6
mordant	adj	pain	0.3	0.5
mordant	adj	clarity	0.4	0.8
savage	adj	violence	1.0	1.0
instinct	sn	intelli-gence	0.5	0.2
instinct	sn	innocence	0.4	0.6
secret	sn	slyness	0.5	0.5
secret	sn	deception	0.5	0.5
prosperous	adj	surfeit	0.5	0.5
rat	sn	disloyalty	0.3	0.9
rat	sn	horror	0.2	0.6
rat	sn	repulsion	0.6	0.7
alarm	vb	fear	0.6	0.5
alarm	vb	warning	0.7	0.7
alarm	vb	excitement	0.8	0.8
portent	sn	promise	0.7	0.9
portent	sn	warning	1.0	0.8
escape	vb	aversion	0.9	0.6
furious	adj	violence	0.8	0.9
furious	adj	anger	1.0	0.8
entertain-ment	sn	pleasure	0.7	0.6
cancel	vb	failure	0.3	0.5
cancel	vb	lack	0.5	0.4
surrealistic	adj	absurdity	0.8	0.5
surrealistic	adj	creation	0.3	0.4
surrealistic	adj	insanity	0.5	0.3
surrealistic	adj	surprise	0.3	0.3
success	sn	success	1.0	0.6
whisper	vb	slyness	0.4	0.5
whisper	vb	slander	0.4	0.4
slander	vb	slander	1.0	0.9
play	vb	creation	0.3	0.3
play	vb	pleasure	0.7	0.5
play	vb	innocence	0.2	0.4
greed	sn	desire	0.6	1.0
greed	sn	greed	1.0	0.7
lust	vb	desire	0.8	0.9
envy	sn	desire	0.7	0.6
envy	sn	greed	0.7	0.6
envy	sn	inferiori-ty	0.4	0.4
envy	sn	lack	0.5	0.5
envy	sn	slyness	0.5	0.6

Fig. 12.3. Entries and Associated Affect Categories with Centralities and Intensities (captured as a figure), as shown in [205]

The authors claim that this 'retrieval-by-similarity' technique could be complemented with the 'filtering-on-intensity technique'. These two, combined together, can become powerful at extracting interesting facts from data. The authors carry one mentioning that combining these techniques with statistical methods would enhance the power of the proposed method. If we were to submit a query, *love, pain* to a text corpus, in search of sentences holding both qualities, "vector space retrieval would return sentences containing many instances of love with a high score, even if there is no mention of pain in them". They claim, on the contrary, that their proposed technique would gives preference to documents having a higher maximal centralities level for the existing categories, irrespectively of their total count in the sentence. Subasic and Huettner believe that this example shows that "in the quantitative approach, we are concerned with finding information that is statistically similar to our query. In the qualitative approach, we are more concerned with a particular qualitative profile of the target information". Clearly, the authors aimed and claimed that their proposed method falls more under the *qualitative profile*.

Sentence/phrase	Centrality Profile	Intensity Profile	Similarity Score	Overall Intensity
fear, anger, grief and pain filled the room	fear/1+anger/1+ pain/1+ sadness/0.9	fear/0.50+ sadness/0.80+ pain/0.70+ anger/0.50	1.0	0.625
there is a lot of anger and violence in media.	anger/1+ violence/1	anger/0.50+ violence/0.60	0.41	0.55
really badly hurt people, badly burnt people	immorality/0.7+pain/0.9 + honesty/0.4+ horror/0.1	immorality/0.50+ pain/0.70+ honesty/0.50+ horror/0.10	0.353	0.45
male characters almost never cried	sadness/0.5+ excitement/0.2	sadness/0.70+ excitement/0.30	0.244	0.5
alarming portents	warning/1.0+ excitement/0.8+ promise/0.7+ fear/0.6	warning/0.75+ excitement/0.80+promise/ 0.90+ fear/0.50	0.1875	0.7375
"masculine" images of danger and aggression	fear/0.4+ violence/0.3+ conflict/0.4+ harm/0.7	fear/0.50+ conflict/0.40+ harm/0.60+ violence/0.30	0.151	0.45
a mordant view of human nature	clarity/0.4+ pain/0.3	clarity/0.80+ pain/0.50	0.14	0.65
violent emotions are dangerous for health	harm/0.70+ strength/0.30+ fear/0.40+ health/1.0+ reasonableness/0.1+ violence/1.0	harm/0.60+ strength/0.20+ fear/0.50+ health/0.50+ reasonableness/0.10+ violence/0.60	0.114	0.42
Death toll in train collision	conflict/0.40+ fear/0.20	conflict/0.70+ fear/0.40	0.09	0.55
Dead person	death/1.0+ horror/0.2+ fear/0.2	death/0.50+ horror/0.20+ fear/0.40	0.078	0.37

Table 12.1. Phrase Retrieval, as reproduced from Subasic and Huettner [205]

12.3.2.2 Case 2: Using fuzzy sets for OM

In 2013, Jusoh and Alfawareh [108] described their approach to using fuzzy sets in opinion mining. Their method includes five steps, those being:

1. Sentence Tokenization
2. SenWord Extraction
3. Assign Positive/Negative Fuzzy Sets
4. Calculate Degree of SenWord
5. Visualization

We will focus on items 2,3 and 4 above, as they are more relevant to the research work we will present in *Part IV* of this report.

- SenWord Extraction: in essence, *SenWord* are words that convey either a positive or a negative message. Jusoh and Alfawareh create a positive lexicon and a negative lexicon, that are sets of SenWord's expressing a positive or a negative connotation, respectively. The work is done manually, using common sense and linguistic knowledge, as per the authors' claim. A real number between [0, 1] is then assigned to a given SenWord. The stronger the connotation of positive, the higher the number would be and the closer to 1.0. For example, *excellent* receives a value of 0.9 whilst *good* gets 0.5. Figure 12.4 shows examples of positive and negative lexicons as presented by [108].
- Assign Positive/Negative Fuzzy Sets: The authors define a fuzzy set of a given review as $F_R = \{(w_i, \mu_R(w_i))$, where w_i is a given word in the lexicon and $\mu_R(w_i)$ is the membership function value of word $w_i\}$. For the following input: 'It is *marvelous*, one of the *best* days I ever lived', the authors claim that there are two SenWords: *marvelous* and *best*. Let us assume as well that in the *positive lexicon* we find pairs (*marvelous*, 0.8) and (*best*, 0.9).

- Calculate Degree of SenWord: the authors use the max operator. Hence, the degree of sentiment of a given opinion (i.e. the one for the opinion expressed above) would be calculated as max{0.8, 0.9} = 0.9.

The method proposed by Jusoh and Alfawareh seems to be straightforward in the sense that it basically creates two lexicons -positive and negative- and then assign scores to each word. It is not completely clear how, but it seems that the lexicons will grow as the computing system processes more information (it looks like new words would get incorporated into the lexicons if they were not part of it already).

Example of a positive lexicon

SenWord	*Positive Value*
Excellent	0.9
Great	0.9
Marvelous	0.8
Wonderful	0.8
Good	0.5
Nice	0.5
Exciting	0.7
Fine	0.3

Example of a negative lexicon

SenWord	*Negative Value*
Worst	0.9
Worse	0.7
Bad	0.7
Ugly	0.9
Useless	0.9
Poor	0.5
Uncreative	0.6

Fig. 12.4. Examples of positive and negative lexicons (captured as a figure), as presented in [108]

12.3.2.3 Case 3: Fuzzy Sets Classification of Chinese Sentiment

In [88], Fu and Wang discuss their ideas on the utilisation of fuzzy sets for sentence-level SA (in Chinese). The authors propose what they call "a fine-to-coarse strategy to estimate sentence sentiment intensity". Fu and Wang define "three fuzzy sets to represent the respective sentiment polarity classes" [88] (negative, positive and neutral) and create membership functions to model the different degrees of opinions in a subjective sentence. Fu and Wang determine polarity at the sentence level "under maximum membership principle" [88]. Let us take an in-depth look at their proposed method.

Sentiment words and morphemes

The authors justify that they must work with morphemes (in Linguistics, a morpheme is is the smallest grammatical unit in a language). It seems that *Chinese sentiment words / items* can be classified as either static polar words or dynamic polar words. For a static word, its polarity will not change, whilst a dynamic word may see its polarity changed by context or domain. According to Fu and Wang, "a pre-compiled dictionary cannot cover all sentiment words in real text, which raises an issue of predicting the polarity of out-of-vocabulary (OOV) sentiment words". In order to deal with this problem, the authors introduce the idea of using sentiment *morphemes*. We will omit from the re-illustration of Table 12.2 below (with some cosmetic changes) the Chinese Characters (Ideograms), but will keep the basic idea behind having sentiment morphemes and more complex sentiment words composed out of sentiment morphemes.

Morpheme-level polarity

The authors explain that word-level polarity is usually determined by main sentiment morphemes within specific sentiment words. As such the words 'undermine' and 'corruption' in the example in Table 12.2 do share the same negative sentiment morpheme, 'fail'. As a consequence, *they do have the same negative orientation*. It is precisely based on this last observation that they decide to use morpheme-level polarity to assisting in predicting the polarity of static sentiment terms, instead of a sentiment lexicon. As there is no dictionary of sentiment morphemes available -or so the authors claim- they propose the (automatic) extraction of sentiment morphemes from existing sentiment lexicon using a chi-square (χ^2), which result in a positive/negative polarity value for a sentiment morpheme if the final chi-square value derived is positive/negative.

Morpheme Types	Sentiment morphemes	Sentiment words composed by sentiment morphemes
Positive morpheme	'beauty'	'exquisite' 'graceful'
	'love'	'like' 'adoration'
Negative morphemes	'dirty'	'pollution' 'corruption'
	'fail'	'corruption' 'undermine'

Table 12.2. Types of Chinese sentiment morphemes as per the authors [88]

Word-level polarity

Word-level polarity is calculated by introducing morpheme-based rules. After a normalization process to bring all values of χ^2, for each sentiment morpheme m into the interval $[-1, 1]$. This chi-square $chi(m)$, which is normalized, is considered to represent the score for the opinion of the sentiment morpheme m. Fu and Wang proceed by claiming that "Thus, we can determine whether a word is a sentiment or not using a simple rule: if a word contains sentiment morphemes, it is a sentiment word. Finally, we can calculate the opinion score of a word w consisting of morphemes m_i, $(1 \leq i \leq 2)$, using the following two rules" [88]:

- If m_1 is a negation (like 'not', 'non-') then $Score(w) = -1$ x $chi(m_2)$.
- If m_1 is not a negation morpheme, then $Score(w) = Sign(chi(m_i))$ x $Max(|chi(m_i)|)$, where $Max(|chi(m_i)|)$ is the largest absolute value among the opinion scores of morphemes within a word w, and the expression $Sign(chi(m_i))$ represents the positive or negative sign of m.

Identifying phrase-level polarity

Fu and Wang apply lexical polarity in order to manage contextual polarity and find out sentiment orientation of a given phrase within an opinionated sentence. They base their approach on the work of Hatzivassiloglou and Wiebe [93] and Turney [212], and they take into consideration four types of structures during the sentiment phrase extraction (see Table 12.3). Contrary to the work of Turney [212], the authors consider "phrases with negations as their initial words. In this way, we can handle the local negation that may reverse polarity" [88]. The authors claim that by using rules similar to those proposed by Bo Pang et al. [102] they can calculate the opinion score of extracted phrases. They use the concept of *increased dynamic polar words* (e.g. high, increase, upgrade) and *decreased dynamic polar words* (e.g. down, reduce, diminish), where $Sign$(Increased dynamic polar word) $= 1$ and $Sign$(Decreased dynamic polar word) $= -1$.

Structure of a Phrase	Examples
Phrases that include an *adjective*	"high success rate"
Phrases including a *verb*	"carefully discuss"
Phrases carrying an *idiom*	"intent to deceive the public"
Phrases starting with a *negation*	"no evidence"

Table 12.3. Structures of opinion phrases, as illustrated by [88]

Sentence Sentiment Classification using fuzzy sets

As per Fu and Wang [88], if X is an unordered set of sentiment opinions (represented by x), then a **a positive-meaning sentiment fuzzy set** $\tilde{P}$ in X can be defined as a set of ordered pairs, namely

$$\tilde{P} = \{(x, \mu_{\tilde{P}}(x)) \,|\, x \in X\},$$

where $\mu_{\tilde{P}}(x)$ denotes the membership function of x in $\tilde{P}$, that maps X to the membership space M. Fu and Wang have chosen to use the *rise semi-trapezoid distribution* [273] as the best function for computing the membership degree of the

positive sentiment fuzzy set

$$\mu_{\tilde{P}}(x) = \begin{cases} 0 & \text{if } x < a; \\ \frac{x-a}{b-a} & \text{if } a \leq x \leq b; \\ 1 & \text{if } x > b. \end{cases} \tag{12.6}$$

where x denotes the opinion score of the sentence under discussion. Parameters a and b seem to be adjustable and they are defined as $a = Min(x_i) + \lambda_1(Max(x_i) - Min(x_i)/k)$ and $b = Min(x_i) + \lambda_2(Max(x_i) - Min(x_i)/k)$. $Max(x_i)$ and $Min(x_i)$ are the maximum and minimum values within X. λ_1, λ_2 and k are parameters that seem to have been set arbitrarily as $\lambda_1 = 5.2$, $\lambda_2 = 5.4$ and $k = 10$.

A neutral meaning sentiment fuzzy set $\tilde{E}$ in X can be stated as a set of ordered pairs

$$\tilde{E} = \{(x, \mu_{\tilde{E}}(x)) \,|\, x \in X\},$$

where

$$\mu_{\tilde{E}}(x) = \begin{cases} 0 & \text{if } x < a; \\ \frac{x-a}{b-a} & \text{if } a \leq x < b; \\ 1 & \text{if } b \leq x < c; \\ \frac{d-x}{d-c} & \text{if } c \leq x < d; \\ 0 & \text{if } x \geq d. \end{cases} \tag{12.7}$$

x represents the opinion score of the sentence under discussion. Parameters a, b, c and d are defined, respectively, as follows:
$a = Min(x_i) + \lambda_1(Max(x_i) - Min(x_i)/k)$, and
$b = Min(x_i) + m_1(Max(x_i) - Min(x_i)/k)$, and
$c = Min(x_i) + m_2(Max(x_i) - Min(x_i)/k)$, and
$d = Min(x_i) + \lambda_2(Max(x_i) - Min(x_i)/k)$.

$Max(x_i)$ and $Min(x_i)$ are the maximum and minimum values within X. λ_1, λ_2, m_1, m_2 and k are parameters that seem to have been set arbitrarily as $\lambda_1 = 5.2$, $\lambda_2 = 5.5$, $m_1 = 5.26$, $m_2 = 5.33$ and $k = 10$.

A *negative sentiment fuzzy set*, $\tilde{N}$ in X, can be defined as a set of ordered pairs, namely

$$\tilde{N} = \{(x, \mu_{\tilde{N}}(x)) \,|\, x \in X\},$$

where

$$\mu_{\tilde{N}}(x) = \begin{cases} 1 & \text{if } x < a; \\ \frac{b-x}{b-a} & \text{if } a \leq x \leq b; \\ 0 & \text{if } x > b. \end{cases} \tag{12.8}$$

x represents the opinion score of the sentence being discussed. Adjustable parameters a and b are represented as:
$a = Min(x_i) + \lambda_1(Max(x_i) - Min(x_i)/k)$, and
$b = Min(x_i) + \lambda_2(Max(x_i) - Min(x_i)/k)$.

$Min(x_i)$ and $Max(x_i)$ are the minimum and maximum values in X. λ_1, λ_2 and k are parameters that seem to have been set arbitrarily as $\lambda_1 = 5.2$, $\lambda_2 = 5.3$ and $k = 10$.

Determining sentence polarity

By utilising the above mentioned membership functions, Fu and Wang can obtain the grade of the membership of a given opinionated sentence in each sentiment fuzzy set, and in turn they can find out as well its polarity under the *principle of*

maximum membership. As such, let $\tilde{A}_1, \tilde{A}_2 \ldots \tilde{A}_n$ be the fuzzy sets of X, $\exists x_0 \in X$, if

$$\tilde{A}_k(x_0) = \max_{1 \leq i \leq n} \{\tilde{A}_i(x_0)\}$$

then x_0 is a membership of the fuzzy set $\tilde{A}_k$.

In our opinion, the work of these researchers establishes a bridge-head for the utilisation of fuzzy sets in computing polarity in sentences that are subjective. It is unclear to us, how the values of the parameters used in defining the membership functions for positive, neutral and negative cases were established (λ_1, m_1, λ_2, m_2 and k). We assume that the values of these parameters were estimated using as basis experimental results. Another aspect to consider is that the methodology used seems to be rather complex on the determination of morpheme-level polarity, but it could be attributed to the use of Chinese language ideograms and predicting polarity of sentiment terms that are not included in the existing vocabulary (aspects like the use of χ^2, etc.). We see the possibility of using in our work some of the ideas behind the three fuzzy sets and membership functions calculations presented in this sub-section, but rather at a conceptual level as it will be apparent later in *Part IV*.

12.3.2.4 Case 4: Sentiment Classification of Customer Reviews based on FL

Nadali et al. [155] present a rather straightforward approach to the topic of applying fuzzy sets to to the SA problem. In essence, their approach is to build a Fuzzy Inference System (FIS) to address the problem at hand, which requires fuzzification and defuzzification steps. The method proposed by Nadali et al., as described in [155], is as follows:

- Part 1: Input and preparation
  1. Review sentence: take the input sentence and apply PoS tagging
  2. Find Opinion words: opinion words are extracted. The following PoS particles are considered by the authors as capable of carrying an opinion: adjectives, adverbs, verbs and nouns.
- Part 2: Fuzzy Logic System
  1. Fuzzify: fuzzification inputs correspond to the four PoS particles identified before. A specific degree is assigned to each opinion word by a human expert (ranging from 0 to 10).
  2. Membership Function (MF) Design: MFs are defined to find membership values for each of the inputs. The authors used a Triangular Membership Function in their proposed method. "Rank of MF is decelerated by human experts" [155]. The linguistic variables utilised (see Fig. 12.5) were created encompassing three levels: Low, Moderate and High. For example, initially the selected PoS particles were assigned *degrees*. For

Fig. 12.5. MF's used to present the linguistic labels, as published in [155]

instance, 'like, 4', 'love, 5', 'good, 3', 'excellent, 6', 'really, 5', 'extremely, 9', 'enjoy, 8' and 'very, 5'. When the MF is applied, we obtain respectively: $\mu(very) = 0.5$, $\mu(like) = 0.4$, $\mu(extremely) = 0.9$, $\mu(good) = 0.3$, and $\mu(enjoy) = 0.8$.

3. Fuzzy Rules Design: In this phase the authors define a number of IF-THEN Rules to address the problem. The IF-THEN rules have been created under the shape of the general form that follows:

$$\text{IF } x_1 \text{ is } H \text{ AND } x_2 \textit{ is } J \text{ THEN } \textit{Orientation} \text{ is } K.$$

$$H, J \in \{\text{Low, Moderate, High}\}$$

$$K \in \{\text{very strong, strong, moderate, very weak, weak}\}$$

See Fig. 12.6 for examples of some of the IF-THEN rules created by the authors.

1	If (Adverb) is (low positive) And (Adjective/Verb) is (low positive) then the orientation is (very weakly positive).
2	If (Adverb) is (Moderate positive) And (Adjective/Verb) is (low positive) then the orientation is (weakly positive).
3	If (Adverb) is (Moderate positive) And (Adjective/Verb) is (Moderate positive) then the orientation is (Moderate positive).
4	If (Adverb) is (High positive) And (Adjective/Verb) is (Moderate positive) then the orientation is (Strong Positive).
5	If (Adverb) is (High positive) And (Adjective/Verb) is (High positive) then the orientation is (Very Strong Positive).

Fig. 12.6. Subset of samples of IF-THEN Rules, as displayed in [155]

4. Defuzzification: They use the Mamdani's center of gravity defuzzification method,

$$y^* = \frac{\int_y \mu(y)\, y \, \mathrm{d}y}{\int_y \mu(y) \, \mathrm{d}y}$$

where y^* is the crisp value; $\mu(y)$ is the MF of the corresponding value y in the previous result.

- Part 3: Final output: as described in the previous item, the crisp value is output. The authors claim that "We define crisp values for each of the output, for example: 'Neutral: 0', Weakly positive: 0.2, very weak positive: 0.4, Moderate positive: 0.6, Strong Positive: 0.8, Very Strong Positive: 1. Based on these numbers we can find the orientation of sentence 1 and 2 as belonging to Very Strong Positive and Moderate Positive".

12.4 Chapter Summary

In this chapter we have presented some basic fuzzy sets concepts and have illustrated their applicability to the SA problem, by sharing specific examples of their utilisation. The fuzzy sets approach to SA is very important to us as we have based our method to identifying sentiment graduality, in the use of fuzzy sets. In the next chapter we will consider options for the aggregation of information in a classification effort, showing as well a number of possible semantics that could be incorporated into an aggregation method.

Chapter 13

Aggregation Methods Fundamentals

> "En una noche del tiempo, un hombre quiso saber quien era Dios. Lanzó su ojo derecho al espacio, y tragó su ojo izquierdo para ver lo que tenía en sus antros."
>
> Gustavo Adolfo Núñez Testa [1958-2013], circa 1973.

In this chapter, we introduce the theoretical basis for the deployment of aggregation approaches rooted in the concepts of achieving consensus and generating a compensatory effect in the aggregation process itself.

There are a number of aggregation methods that could be utilised in a number of disciplines, domains and scenarios. The ones being presented in this chapter correspond to the aggregation methods we have put at use through some of the proposed solutions that were devised in our research. Mainly, we will cover Fuzzy majority aggregation through the use of Induced Ordered Weighted Averaging (IOWA) Operators and Uninorm Aggregation. The methods we are describing in this chapter will be utilised and further explained in Chapter 16.

13.1 OWA Operators

Usually, the first step taken in a group decision-making resolution process is that of aggregating the information from which to derive a group solution to the problem. Yager's OWA operator [256] has been proved to be extremely useful in these decision making problems because it allows to implement the concept of *fuzzy majority* [251].

Definition 13.1 (OWA Operator)**.** : *An OWA operator of dimension n is a function $F : \mathbb{R}^n \to \mathbb{R}$, that has associated a set of weights or weighting vector $W = (w_1, \ldots, w_n)$ to it, so that $w_i \in [0,1]$ and $\sum_{i=1}^{n} w_i = 1$, with the following expression:*

$$F(a_1, \ldots, a_n) = \sum_{i=1}^{n} w_i \cdot a_{\sigma(i)},$$

being $\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$ a permutation such that $a_{\sigma(i)} \geq a_{\sigma(i+1)}, \forall i = 1, \ldots, n-1$ (i.e., $a_{\sigma(i)}$ is the i-th highest value in the set $\{a_1, \ldots, a_n\}$).

If B is the vector whose components are the ordered arguments values, $b_i = a_{\sigma(i)}$, then:

$$F(a_1, a_2, \ldots, a_n) = W^T B \tag{13.1}$$

A point that deserve attention in the definition of the OWA operator is how to obtain the associated weighting vector. In [256], Yager proposed two ways to obtain W. The first approach he introduced is the use of some kind of learning mechanism using some sample data; and the second approach he described is that of trying to give some semantics or meaning to the weights. The latter case allowed applications in the area of quantifier guided aggregations [251] because weights are derived "from a functional form of the linguistic quantifier" [168]. This approach would be favourable for

the problem we are considering because it allows the aforementioned implementation of the concept of 'fuzzy majority'. This will be exploited later on in our proposed consensus approach to the SA problem driven by support-based IOWA majority presented in Chapter 16. Notice that the average operator is a particular type of OWA operator with weighting vector $W_{Average} = [1/n, 1/n, \ldots, 1/n]$.

According to Pasi and Yager, let $Q : [0,1] \rightarrow [0,1]$ be a function such that $Q(0) = 0$, $Q(1) = 1$, and $Q(x) \geq Q(y)$ for $x > y$ corresponding to a fuzzy set representation of a proportional monotone quantifier. Then, for a given value $x \in [0,1]$, the value $Q(x)$ corresponds to the *degree* to which *x satisfies* the fuzzy concept being represented by the *quantifier* [168]. Based on function Q, the elements of the OWA weighting vector are determined in the following way [168]:

$$w_i = Q\left(\frac{i}{n}\right) - Q\left(\frac{i-1}{n}\right) \tag{13.2}$$

Hence, w_i represents the increase of satisfaction in getting i with respect to $(i-1)$ criteria satisfied.

Some examples of linguistic quantifiers, depicted in Fig. 13.1, are *"at least half", "most of"* and *"as many as possible"*, which can be represented by the following function

$$Q(r) = \begin{cases} 0 & \text{if } 0 \leq r < a \\ \frac{r-a}{b-a} & \text{if } a \leq r \leq b \\ 1 & \text{if } b < r \leq 1 \end{cases} \tag{13.3}$$

utilising the values $(0, 0.5)$, $(0.3, 0.8)$ and $(0.5, 1)$ for (a, b), respectively [110].

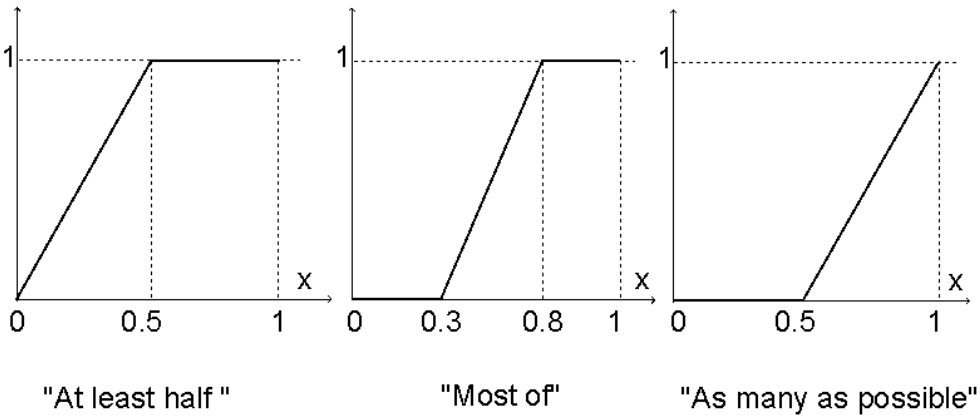


Fig. 13.1. Linguistic quantifiers *"at least half", "most of"* and *"as many as possible"*

Alternative representations for the concept of fuzzy majority can be found in the literature. For example, Yager in [251] considered the parameterized family quantifiers $Q(r) = r^a$ $(a \geq 0)$ as a possible representation. This family of functions guarantees that [54]: (i) all the experts contribute to the final aggregated value (strict monotonicity property), and (ii) associates, when $a \in [0,1]$, higher weight values to the aggregated values with associated higher importance values (concavity property).

The degree of *orness* of an OWA aggregation operator expresses its closeness to the OR behaviour, which is defined as

$$orness(\mathbf{W}) = \left(\frac{1}{n-1}\right) \sum_{j=1}^{n} ((n-j) * w_j) \tag{13.4}$$

The OWA operator F^* with the weighting vector W^* defined as $[1, 0, \ldots, 0]$ corresponds to the OR operator (i.e., the *max*), in which case, $orness(W_*) = 1$. Similarly, the OWA operator with the weighting vector W_* defined as $[0, \ldots, 0, 1]$ corresponds to the AND operator (i.e., the *min*), in which case, $orness(W_*) = 0$. It is clear that the orness of the Average operator, i.e. the OWA operator with equal weighting vector components, is 0.5. It is important to mention some key properties of OWA operators [256, 257].

Property 1. *For any OWA operator F*

$$F_*(a_1, a_2, \ldots, a_n) \leq F(a_1, a_2, \ldots, a_n) \leq F^*(a_1, a_2, \ldots, a_n).$$

Property 2. *(Commutative). Let $\langle a_1, a_2, \ldots, a_n \rangle$ be a bag of aggregates and let $\langle d_1, d_2, \ldots, d_n \rangle$ be a permutation of the a_i's. Then for any OWA operator*

$$F(a_1, a_2, \ldots, a_n) = F(d_1, d_2, \ldots, d_n)$$

Property 3. *(Monotonicity). Assume a_i and c_i are collection of aggregates, $i = 1, \ldots, n$, such that for each i, $c_i \leq a_i$, then*

$$F(c_1, c_2, \ldots, c_n) \leq F(a_1, a_2, \ldots, a_n),$$

where F is any OWA operator.

Property 4. *(Idempotency). If $a_i = a$ for all $i = 1, \ldots, n$ then for any OWA operator*

$$F(a, a, \ldots, a) = a.$$

These properties show that OWA operators comply with the properties that are expected in an averaging operator. Another key measure of OWA operators is the *dispersion*, defined as:

$$Disp(W) = -\sum_{i=1}^{n} w_i \,.\, \ln(w_i).$$

$Disp(W)$ measures the degree to which all aggregates are used equally in the resulting final aggregation [169].

Notice that for $orness(W) \in [0, 1]$, the nearer W is to a logical OR, the closer its value is to 1; whilst the nearer it is to a logical AND, the closer its value is to 0. As per [169], in general, "an OWA operator with much of non-zero weights near to the top will be an *or-like* operator", $orness(W) \geq 0.5$, "and when much of the weights are non-zero near to the bottom, the OWA operator will be *and-like*", with $andness(W) = 1 - orness(W)$. As per the discussion presented in [169], "the following theorem shows that as we move weight up the vector we increase the $orness(W)$, while moving weight down causes us to decrease $orness(W)$".

Theorem 13.1. *Assume W and W′ are two n-dimensional OWA vectors such that*

$$W = [w_1, w_2, \ldots w_n]^T, \quad W' = \left[w_1, \ldots, (w_j + \epsilon), \ldots, (w_k - \epsilon), \ldots, w_n\right]^T,$$

where $\epsilon > 0$, $j < k$.

Then $orness(W') > orness(W)$. According to Yager [256], OWA operators are among the most commonly utilised operators in multi-criteria decision-making and aggregation in situation where only some portion of the criteria must be satisfied. However, as we will present in the next Section 13.1.1 and in Chapter 16, the semantic of the OWA's aggregation process we have shown does not provide a good representation of the *majority concept*. For that, we will introduce a generalisation of the OWA operator with a specific semantic in the aggregation process.

13.1.1 IOWA Operators

Mitchell and Estrakh in [150] described a modified OWA operator in which the input arguments are not re-arranged according to their values but rather using a function of the arguments. Inspired by this work, Yager and Filev introduced in [258] a more general type of OWA operator, which they named the Induced OWA (IOWA) operator:

Definition 13.2 (IOWA Operator)**.** *An IOWA operator of dimension n is a mapping $I - F \colon (\mathbb{R} \times \mathbb{R})^n \longrightarrow \mathbb{R}$, which has an associated set of weights $W = (w_1, \cdots, w_n)$ to it, so that $w_i \in [0,\ 1]$, $\sum_{i=1}^{n} w_i = 1$,*

$$I - F\left(\langle u_1, a_1 \rangle, \ldots, \langle u_n, a_n \rangle\right) = \sum_{i=1}^{n} w_i \cdot a_{\sigma(i)},$$

and $\sigma: \{1,\ldots,n\} \longrightarrow \{1,\ldots,n\}$ *is a permutation function such that* $u_{\sigma(i)} \geq u_{\sigma(i+1)}$, $\forall i = 1,\ldots,n-1$.

In the above definition the reordering of the set of values to aggregate, $\{a_1,\ldots,a_n\}$, is induced by the reordering of the set of values $\{u_1,\ldots,u_n\}$ associated to them, which is based upon their magnitude. Yager and Filev called the vector of values $(u_1,\ldots,u_n)$ the order inducing vector and $\{a_1,\ldots,a_n\}$ the values of the argument variable [252, 258]. Thus, the main difference between the OWA operator and the IOWA operator is the reordering step of the argument variable. In the case of OWA operator the reordering is based upon the magnitude of the values to be aggregated, while in the case of IOWA operator an order inducing vector is used as the criterion to induce that reordering of the values to aggregate. Obviously, an immediate consequence of definition 13.2 is that if the order inducing vector components coincide with the argument values then the IOWA operator reduces to the OWA operator. In fact, the OWA operator as well as the weighted average (WA) operator are included in the more general class of IOWA operators, which means that the IOWA operators allow to take control of the aggregation stage of any multi-criteria decision making problem in the sense that importance can be given to the magnitude of the values to be aggregated as the OWA operators do or to the information sources as the WA operators do. In fact, the IOWA operator, in our opinion, can play a significant role in the proposed Hybrid Solution to the SA problem as elaborated in Chapter 16.

13.2 Uninorms

Aggregation operators are usually classified into one of the following three categories:

(i) **Conjunctive operators** like the family of t-norm operators, which has the minimum operator as its largest element. These operators behave like a logical "and"

(ii) **Disjunctive operators** like the family of t-conorm operators. These operators are the "dual" of conjunctive operators, and they behave like a logical "or". The maximum operator is the smallest of all t-norms operators.

(iii) **Compensative operators** are located between the minimum and the maximum operators, and consequently are neither conjunctive nor disjunctive. These type of operators are known as "averaging operator" and they are widely used in multi-criteria decision making problems. The arithmetic mean, the weighted mean and the ordered weighted averaging (OWA) operators are representative examples of this class.

It is worth mention *the family of uninorm operators* [255] as it does not belong fully to any of the three classes described above. Indeed, a uninorm operator, U, is defined as a is a mapping $U: [0,1]^2 \longrightarrow [0,1]$ satisfying the properties:

1. Commutativity: $U(x,y) = U(y,x)$
2. Monotonicity: $U(x_1,y_1) \geq U(x_2,y_2)$ if $x_1 \geq x_2$ and $y_1 \geq y_2$
3. Associativity: $U(x,U(y,z)) = U(U(x,y),z)$
4. Identity element: $\exists\, e \in [0,1] :\ \forall\ x \in [0,1],\ U(x,e) = x$

Uninorm, t-norm and t-conorm operators share the commutativity, associativity and monotonicity properties. However, the set of uninorm operators has both the set of t-norm operators and the set of t-conorm operators as its subsets. Indeed, a uninorm operator with "$e = 1$" becomes a t-norm operator; while a uninorm operator with "$e = 0$" becomes a t-conorm operator. In general, a uninorm operator with identity element $e \in]0,1[$ behaves like (i) a t-norm operator when all aggregated values are below e; (ii) a t-conorm operator when all aggregated values are above e; (iii) a compensative operator otherwise.

13.3 Chapter Summary

In this chapter we had illustrated the importance of efficient aggregation techniques. OWA and *IOWA* operators were described, as well as uninorms, in particular the special type of cross-ratio uninorm operators. In Part IV, we will discuss in details our proposed solution, which encompasses three models related to the SA classification problem:

- A Hybrid Approach to the SA Problem at the Sentence Level (Chapter 14).

- A Model Extension using Aggregation by Uninorm (Chapter 15).
- A Model Extension utilising Aggregation by Consensus (Chapter 16).

The three components of our proposed solution to the SA problem at the sentence level will be further elaborated in *Part IV*, in the chapters mentioned in the itemisation above. In these chapters we will include as well the experimental evaluation of our proposed solution.

Part IV

PROPOSED SOLUTION & EXPERIMENTAL RESULTS

Chapter 14

A Hybrid Approach to the SA Problem at the Sentence Level

> "According to the Latin and Greek classics, the lamias lived in Africa. The top part of their bodies was that of a beautiful woman; the bottom, took the shape of a serpent. They had a remote divine origin, as they were the result of one of many love affairs of Zeus.".
> "Según los clásicos latinos y griegos, las lamias habitaban en Africa. De la cintura para arriba su forma era la de una hermosa mujer; más abajo la de una sierpe. Su remoto origen era divino; procedían de uno de los muchos amores de Zeus.".
>
> Jorge Luis Borges [32]

A significant part of the content of this chapter was used in articles published by the author. See references [10, 11, 12, 13, 14, 15, 16, 17, 18]. In this chapter we will describe in detail our proposed solution to the sentiment analysis problem at the sentence level.

14.1 Hybrid Standard Classification (HSC) Method

Let us clarify first what we mean by utilising a 'hybrid approach' that is key to our proposed solution. Our intention is to manage *hybrid concepts* at two different levels: (a) the methods employed by the sentiment classifiers, and (b) the techniques used to build key components in our approach, like the creation and population of the sentiment/opinion lexicon, and the word dictionaries. The following paragraphs will discuss the different components of our proposed hybrid solution. A graphic depiction of our proposed system is provided in Fig. 14.1.

14.1.1 Component 1: the sentiment/opinion lexicon

Liu compiled an opinion lexicon that "does include a list of positive and negative opinion words or sentiment words for English (around 6,800 words) [...] compiled over many years starting from [their] first paper [102]" (`http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#lexicon`).

These opinion lexicon words will be used as a starting point in our proposed solution to the SA problem and they will be enriched in a number of ways, including a new structure and organisation more adequate for the research approach proposed here. Part of the reasoning behind using Liu's lexicon is to re-use data resulting from an existing good quality example of words compilation and as a point of *commonality* with previous research efforts for benchmarking purposes. In generating our own sentiment/opinion lexicon we have taken the following actions:

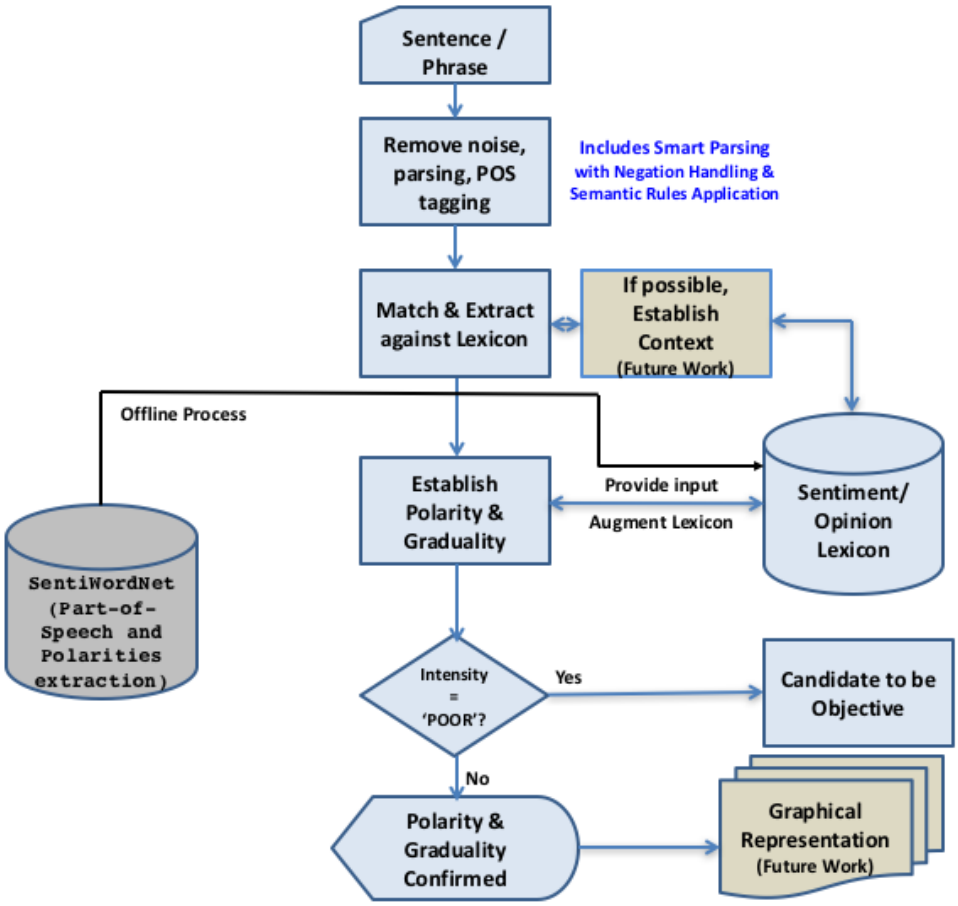


Fig. 14.1. View of our proposed hybrid approach

1. We have utilised, as a starting point, the *opinion-conveying-words* that are part of the opinion lexicon used by Hu and Liu in [102]. These words correspond *only* to four elements of part-of-speech (PoS) that have been proved to be capable of delivering opinions [92, 94, 111, 234]: nouns, verbs, adjectives and adverbs.
2. We have used SentiWordNet [78, 79, 80], which extends some of the functionality of WordNet [83, 147], as a source of *polarity or valence scores* for words that were originally in Hu and Liu's list. As such, we have taken the words supplied by Hu and Liu, and looked them up in SentiWordNet. For those terms matching, the Positive and Negative scores available in SentiWordNet have been extracted and entries in our lexicon have been created combining the words from Hu and Liu's list and the semantic attributes present in SentiWordNet.

In terms of the characteristics of the polarity scores extracted from SentiWordNet, it is important to keep in mind that the polarity scores belong in the interval $[0,1]$. Hence

$$0 \leq PositiveScore, NegativeScore, ObjectivityScore \leq 1$$

$$0 \leq (PositiveScore + NegativeScore) \leq 1$$

$$ObjectivityScore = 1 - (PositiveScore + NegativeScore)$$

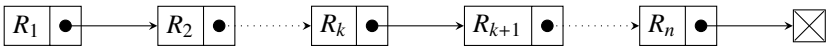
As such, when the sum of $PositiveScore$ (PSC) and $NegativeScore$ (NSC) is equal to 1 for a given word $Word_k$, then the term $Word_k$ is fully opinionated, as opposed to the case when the addition of these two scores is zero, in which case the term $Word_k$ is fully Neutral or Objective. The *Objectivity Score* (COBJ) can be seen as a value of ambiguity/hesitancy, as it is the difference between 1 and the classification of a word as a negative/positive carrier of meaning. Thus, if PSC and NSC add, for example, to 0.9, then there is 0.1 points for the given word to occupy a semantic neutral space or hesitancy.

Not every word in Liu's opinion lexicon is present in SentiWordNet. Hence, for those absent words we have chosen

to keep them in the new opinion lexicon, but they do not have polarity scores associated nor the proper PoS tag. They have been flagged in a special way so they can be recognised and enriched once the required information becomes available. Entries in the sentiment lexicon have the following structure:

$$R = (Word, SOL, PoS, PSC, NSC, COBJ, VDX, UPDC)$$

then we get the following graphical representation for elements in the sentiment lexicon ($n = length(lexicon)$ = number of words in the lexicon):



A full description of the components of our Sentiment/Opinion Lexicon follows:

Word: word in the lexicon (entries).

SOL: semantic orientation label (pos/neg); inherited from Hu & Liu list [102].

PoS: part of speech (n=noun; v=verb; a=adjective; r=adverb; s=adjective satellite).

PSC: Positive Score as taken from SentiWordNet [79].

NSC: Negative Score as taken from SentiWordNet [79].

COBJ: Calculated Objectivity Score.

VDX: Versioning index for identifying/managing synonyms (future use).

UPDC: Update Counter to keep track of every time a given entry in the lexicon is updated.

Section 14.1.4.1 will explain the mechanics of how the sentiment lexicon is utilised.

Notice that our prototype has been built as a proof of concept tool, and not *yet* as a finished software product. Several of the programming constructs and data structures used correspond to native features of the programming language used for creating the prototype: Scheme [73, 207], a dialect of LISP [145]. As part of future work, we intend to port the code to a member of the family of the C programming language. Then, we will focus on algorithm efficiency by using the appropriate data structures (e.g. hash map instead of list data type) and effective programming techniques.

14.1.2 Component 2: semantic rules (SR)

In a classical SA approach with linguistic content semantic rules are utilised, as they assist in modelling the SA problem in a more rigorous fashion. In addition to the most common rules, a number of authors, among them [155, 205, 248], have pointed out the fact that having rules for negation handling and to deal with the use of specific PoS particles, like 'but, despite, unless, ...' could positively affect the final outcome of a classification exercise. Thus, some rule strategy is needed to be put in place as the order of the different PoS play a role in the semantic of a sentence. Researchers have been, through time, improving the quality of these semantic rules so that they are more encompassing of the possible cases that must be managed. These research efforts are summarised by Xie et al. in [248], which includes a full presentation of their semantic rules approach. Up to certain extent, the semantic rules devised in the current hybrid proposed system are based on those presented by Xie et al. and as such a subset of the rules the aforementioned authors presented is utilised with the incorporation of new rules. We have followed the same naming convention for rules utilised by Xie et al. (R1, R2, ..., R13) with the addition of a super-index (Rk^{HSC}) to represent the rules actually utilised in the proposed method. Gaps in the sequential numbering utilised by Xie et al. represent rules that have not been implemented (R2, R4, R5, R8 and R9). The semantic rules utilised in the proposed method are displayed in Table 14.1 and Table 14.2.

Despite the apparent completeness of existing semantic rules by Xie et al., we have incorporated *two new rules* for managing particular PoS particles that were not included in the original set of rules provided in [248]: the particle **while** and the particle **however**. The process of adding these rules was the result of observation and experimentation. We first looked at the cases of sentences in the datasets and examined whether the existing semantic rules were capable of processing the data. Secondly, we look for syntactic structure patterns in the data, and realised that treating the particles

Rule	Semantic Rules	Example
$R1^{HSC}$	Polarity (not var_k) = -Polarity (var_k)	'not *bad*.'
$R3^{HSC}$	Polarity (NP_1 VP_1) = *Compose* (NP_1, VP_1)	'*Crime* has decreased.'
$R6^{HSC}$	Polarity (ADJ to VP_1) = *Compose* (ADJ, VP_1)	'*Unlikely* to destroy the planet.'
$R7^{HSC}$	Polarity (VP_1 NP_1) = *Compose* (VP_1, NP_1)	'*Destroyed* terrorism.'
$R10^{HSC}$	Polarity (not as ADJ as NP) = -Polarity (ADJ)	'That wasn't as *bad* as the original.'
$R11^{HSC}$	If sentence contains "but", disregard all previous sentiment and only take the sentiment of the part after "but".	'And I've never liked that director, *but* I loved this movie.'
$R12^{HSC}$	If sentence contains "despite", only take the sentiment of the part before "despite".	'I love the movie, *despite* the fact that I hate that director.'
$R13^{HSC}$	If sentence contains "unless", and "unless" is followed by a negative clause, disregard the "unless" clause.	'Everyone likes the video *unless* he is a sociopath.'

Table 14.1. Semantic rules actually implemented in our Hybrid Approach (HSC)

Compose Functions Revised	Algorithms
Compose (arg1, arg2)	1. Return -Polarity(*arg2*) if *arg1* is negation.
	2. Return Polarity(*arg1*) if (Polarity(*arg1*) = Polarity(*arg2*).
	3. Otherwise, return the majority term polarity in *arg1* and *arg2*.

Table 14.2. Compose function implemented in HSC

Rule	Semantic Rules	Example
$R14^{HSC}$ New	If sentence contains "while", disregard the sentence following the 'while' and take the sentiment only of the sentence that follows the one after the 'while'.	'*While* they did their best, the team played a horrible game.'
$R15^{HSC}$ New	If sentence contains "however", disregard the sentence preceding the 'however' and take the sentiment only of the sentence that follows the 'however'.	'The film counted with good actors. *However*, the plot was very poor.'

Table 14.3. New semantic rules extending those presented by Xie et al. in [248]

"while" and "however" as special elements would result in better semantic orientation computation performance. The new rules are given in Table 14.3.

Section 14.1.4.1 will explain the mechanics of how the semantic rules are put at work.

14.1.2.1 Negation effects

Negation can play a major role in identifying subjectivity polarity, and more generally, meaning. According to the well-known researcher Christopher Potts, Stanford University (`http://sentiment.christopherpotts.net/lingstruc.html`), "sentiment words behave very differently when under the semantic scope of negation". Dr. Potts notices that the so-called 'Weak' (mild) words such as *good* and *bad* behave like their opposites when negated (bad = not good, good = not bad), whilst 'Strong' (intense) words like *superb* and *terrible* have very general meanings under negation. According to Potts [177] "not superb is consistent with everything from horrible to just-shy-of-superb, and different lexical items for different senses. These observations suggest that it would be difficult to have a general *a priori* rule for how to handle negation. It does not just turn good to bad and bad to good. Its effects depend on the words being negated. An additional challenge for negation is that its expression is lexically diverse and its influences are far-reaching (syntactically speaking)". The method that Dr. Potts seems to favour for approximating the effects of negation is due to Das and Chen [70] and Pang, Lee, and Vaithyanathan [165]. When this method is incorporated at the tokenization level, the negation problem is relatively well managed. Let us look at an example:

Example 1. *I don't think I will enjoy it: it might be too spicy.*

As per the negation handling technique just mentioned, all words between the negation particle don't and the colon (:) would be tagged with the suffix '_NEG', clearly defining the scope of the negation. All words after the colon (:) would not be tagged at all.

Notice that even long-distance effects can be effectively managed. In our proposed approach, we have *chosen* to apply this *smart tokenization* strategy. There are good reasons for that. First of all, it saves us time as the scope of negation is defined early on, and if a polarity inversion is required, it can be done at tokenization time. Secondly, if a part of a sentence is identified as one that will not contribute to the final semantic orientation, then such part of the sentence can be discarded at this point, minimising the effort required at sentiment computing time.

14.1.3 Component 3: fuzzy sets approach to the SA problem

We have already established that we rely on an opinion lexicon that has been developed using a number of techniques and that started out with opinion-conveying words that were compiled by linguists and other scientists interested in the SA problem. In this sub-section we address the rest of components necessary to be able to classify sentences into *Positive* or *Negative* and, in addition, qualify the strength of the associated polarity. In order to do that we must address the following:

- Describe and construct the *fuzzy methodology* that will be utilised in this effort.
- Describe the fuzzy granulation, i.e. the linguistic discrimination, that will be implemented to represent the subjective classification of sentences into positive or negative.
- Provide the logic necessary, in combination with the lexicon and the fuzzy sets already mentioned, to address the classification problem at hand.
- Describe the mechanics behind the whole process as we incorporate the use of fuzzy sets components.

14.1.3.1 Basic concepts on perceptions and linguistic variables for polarity intensity

When we refer to Natural Languages (English in our case) it is clear that humans have developed the ability to classify objects, without the need to produce an actual measurement. When we say that someone or something is *slightly* large, *very* large, or *not very* large, we all understand the message. However, we have not measured or produced metrics for the object we are referring to. We continually use perceptions in the context of multiple events. According to Zadeh [266]: "reflecting the bounded ability of the human brain to resolve detail, perceptions are intrinsically imprecise. In more concrete terms, perceptions are f-granular, meaning that (1) the boundaries of perceived classes are unsharp and (2) the values of attributes are granulated, with a granule being a clump of values (points, objects) drawn together by indistinguishability, similarity, proximity, and function" (see Fig. 14.2 for a re-illustration of the graphic originally published by Zadeh in [267]). In [267], Zadeh continues, by saying that "a granule may be crisp or fuzzy, depending on whether its boundaries are or are not sharply defined. For example, age may be granulated crisply into years and granulated fuzzily into fuzzy intervals labeled very young, middle-aged, old and very old." Fig. 14.3 re-illustrates the graphical representation of the latter idea as originally presented by Zadeh in [266]. When it comes to the *Theory of Perceptions*, Zadeh's contribution is unique [266, 267]. Additionally, in 1973 Zadeh introduced the concept of linguistic variables: "a variable whose values are words instead of numbers" [262].

When deciding which *linguistic variables* to use in modelling our problem, we came to the realisation that the *intensity* or *degree of polarity* with which the grade of positivity or negativity of a sentence X could be understood corresponds to a *perception*. More specifically, the perception P_X that a given person Y has about how positive or negative a sentence X might be. A sentence could either be *Negative* or *Positive*, and then again 'Most Positive' or 'Very Positive', or 'Most Negative' or 'Very Negative', and so on. Based on the definitions and concepts provided above by Zadeh, a fuzzy granulation of positive/negative sentiment using fuzzy intervals is considered appropriate. According to G.A. Miller [148], *7 plus or minus 2*, is the effective number of categories that a subject (individual or person) can maintain. In our case, we have chosen a conservative approach and have devised 5 labels (7 minus 2), symmetrically distributed in the domain $[0,1]$. Additionally, our choice of trapezoidal function obeys to the fact that it generalises a triangular function and we have aimed for both: more generality and for more than one value at the top of every category. A trapezoidal membership function (MF), as shown in Fig. 14.4, is usually represented by the following 4-tuple (a,b,c,d).

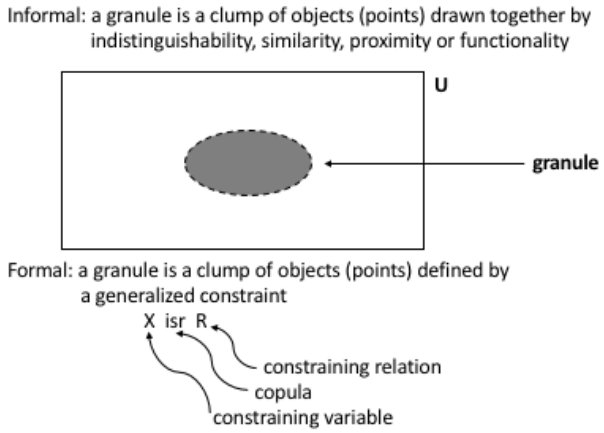


Fig. 14.2. The Concept of a Granule as presented by Zadeh

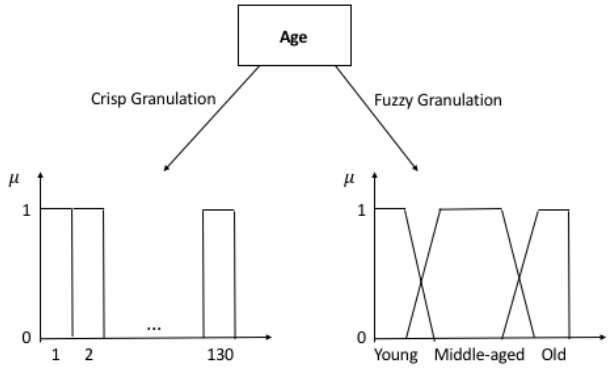
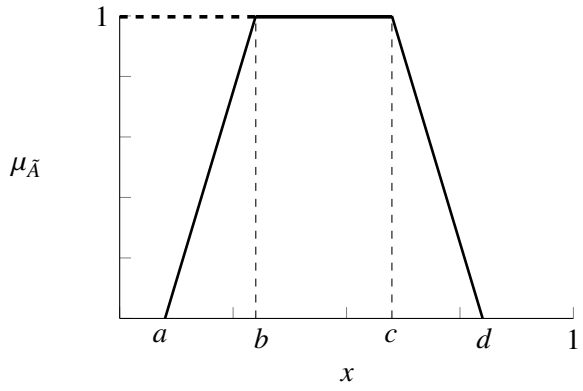


Fig. 14.3. Crisp Granulation and Fuzzy Granulation as introduced by Zadeh



$$\mu_{\tilde{A}}(x) = \begin{cases} 0 & \text{if} \quad x \le a; \\ \frac{x-a}{b-a} & \text{if} \quad a \le x \le b; \\ 1 & \text{if} \quad b \le x \le c; \\ \frac{d-x}{d-c} & \text{if} \quad c \le x \le d; \\ 0 & \text{if} \quad d \le x. \end{cases}$$

Fig. 14.4. Trapezoidal membership function

Specifically, the following granules on the perception of the *positivity* or *negativity* of a given sentence X are suggested: $G = \{Poor; Slight; Moderate; Very; Most\}$, with the following 4-tuples:

- MF (Poor): (0, 0, 0.050, 0.150)
- MF (Slight): (0.050, 0.150, 0.250, 0.350)
- MF (Moderate): (0.250, 0.350, 0.650, 0.750)
- MF (Very): (0.650, 0.750, 0.850, 0.950)
- MF (Most): (0.850, 0.950, 1, 1)

Thus, the *intensity* associated with the semantic positive/negative scores for words occupies a certain fuzzy interval as Fig. 14.5 illustrates. Section 14.2 will explain the mechanics of how the fuzzy sets described are utilised in determining the graduality of the intensity of polarity.

Calculating the level of intensity on the polarity of a sentence is advantageous in itself, as now it can be determined how strong or weak a given positive/negative sentiment might be in natural language. Hence, we are able to say that the sentiment towards a specific sentence is *moderately* positive/negative, *poorly* positive/negative, *most* positive/negative, etc. as per the linguistic labels we have already defined in section 14.1.3.1. Indeed, linguistic polarity intensity could be amenable to be further processed via the *computing with words* methodology introduced by Zadeh in [267], enabling computing with sentiments to be realised in practice.

Natural Languages are a prime example of ambiguity, hidden meanings and multiple interpretations. In future, the proposed method could be taken to a next level, which should include the ability to incorporate approximate reasoning,

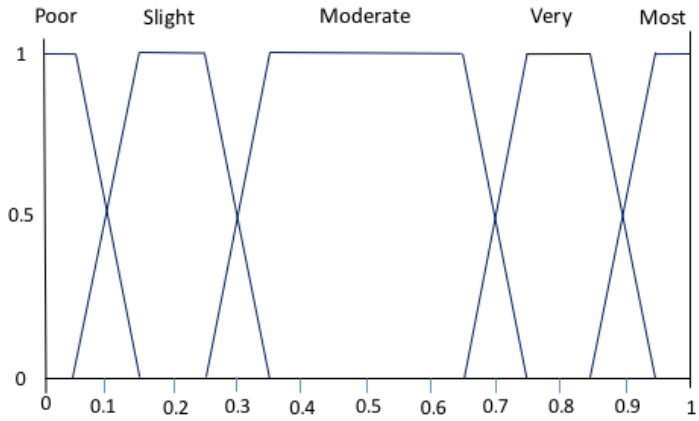


Fig. 14.5. Linguistic variables, fuzzy granulation and trapezoidal membership functions

via Fuzzy Logic, by bringing in the capability of computing with sentiments carried by words and sentences. Potential application around social media, such as product review sites, seems an obvious choice because the sentiment around a specific product based on the linguistic labels that the reviewers have given to the product (good, bad, acceptable, etc.) are possible to be computed. In this case, the entities manipulated to calculate the accumulated or aggregated sentiment would be **words** as opposed to **numbers**. Let us keep in mind that a sentiment in social media is typically expressed *in words and not in numbers*; at least for the regular user. In Fig. 14.6, a word is a label of a fuzzy set and those example labels are *poor, most and slight*. The computation of the aggregated sentiment is performed by directly manipulating the sentiment labels provided by each reviewer. In addition, in the presence of a proper fuzzy logic system, deductions can also be made out of facts expressed in words or determined via the SA approach presented here. In conjunction with social network analysis (SNA) [226, 242, 244, 247], it could be possible for a company, for marketing purposes for example, to identify the most influential nodes in a network that have a *very* or *most* positive sentiment towards a particular product [170, 171, 172].

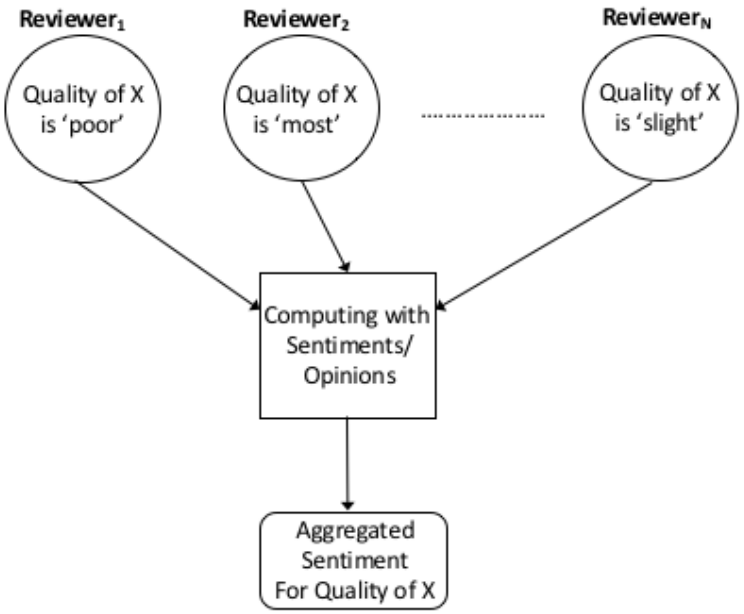


Fig. 14.6. Computing with Sentiments - General Diagram

14.1.4 The hybrid approach (HSC) and Its process

In this section we will describe how all the pieces fall into place in our proposed hybrid method in order to calculate both the sentiment polarity and the intensity of such polarity. Our approach consists of 2-steps, which will be described in the next two sub-sections.

14.1.4.1 Calculating the polarity of sentiments in sentences

There are several tasks that must be executed in strict order in order to determine the polarity of sentences. Every intermediate step has an outcome that is consumed by the next step. Briefly, the tasks are as follows:

1. Tokenization, error cleansing, PoS tagging and smart parsing. The semantic rules, as per section 14.1.2, are applied at tokenization/tagging time and, when applicable, sub-sentences may be discarded at that point (e.g. if the particle 'but' is present, the sub-sentence before the particle 'but' will be discarded). In addition, if a sentence is made of two or more sub-sentences, the proper tagging is performed so that at *interpretation time* the overall polarity is calculated as per the appropriate composition rule (Table 14.2). This step would imply changing the polarity of a given word-sentiment-carrying particle if such a particle is *negated*.
2. The resulting essential particles that convey sentiment/opinion (adjective, nouns, verbs and adverbs) are looked up in the sentiment lexicon bringing across the semantic properties (PoSs and polarity scores) of the term matched.
3. Words that are not in the Opinion Lexicon are tagged as such. Details on how these words are treated are provided in Section 14.3.2.
4. The semantic orientation (SOR) of each sentence is calculated following the process described after this itemised list.
5. After processing a given sentence, those words without a pos/neg label are treated as an exception. This situation happens when the word in question was not in SentiWordNet or it was present, but there were no polarity scores available. More details are given in Section 14.3.1.
6. The overall sentiment of the sentence is produced, and if indicated as such by the semantic rules, a composition is performed for those sentences made of a collection of sub-sentences to derive its compounded semantic orientation (CSO) based on its sub-sentence SOR values. The actual process of computing a sentence SOR is addressed in the paragraph below.

14.1.4.1.1 Computing a sentence SOR. During the actual semantic orientation calculation, both the pos/neg label associated with the words in the lexicon and their respective polarity scores, are taken into consideration. The proposed system performs word counting of both orientations (neg/pos) for every sentence.

If *count*(positive words) > *count*(negative words)
then [the sentence is classified as 'positive'], hence SOR = 'Positive'

If *count*(positive words) < *count*(negative words)
then [the sentence is classified as 'negative'], hence SOR = 'Negative'

If *count*(positive words) = *count*(negative words)
then [There is a tie. Follow alternative process], hence SOR = Table 14.4 result.

Ties are resolved using a three-level stratified algorithm, as displayed in Table 14.4. The different strata shown are mutually exclusive, and every step is executed *only if* the previous step does not resolve the existing tie. As the positive/negative IP values in our lexicon range is $[0,1]$, the semantic orientation calculation requires: (i) the Positive/Negative label in our lexicon (SOL) and (ii) the positive/negative IP values in our lexicon.

If a sentence S is made of n sub-sentences $(S_1, S_2, \ldots, S_n)$, then the CSO of the full paragraph/sentence is calculated by SOR sub-sentence counting.

Strata	Task
Stratus 1	The polarity scores or intensity of polarity (IP) are reviewed and the highest value (among negative and positive words) wins
Stratus 2	There is a hierarchy of importance around the PoS particles to which the words in a sentence belong. The aforementioned hierarchy, from most *influential* to least influential is: (i) adjectives, (ii) adverbs, (iii) verbs and (iv) nouns. If the previous step fails to produce a classification, the hierarchy just described is used and a higher priority is assigned to the IP values of adjectives, followed by adverbs, verbs and nouns
Stratus 3	If the two previous steps fail, we examine our dictionary and search for the participant words; we extract the frequencies with which each word has appeared in a sentence with a specific polarity (pos/neg); the polarity associated to the highest value wins

Table 14.4. Stratified Algorithm for Tie Breaks

1. **If** *count*(Positive SOR Sentences) > *count*(Negative SOR Sentences)
   then $CSO_{(S_1,S_2,...,S_n)}$ = Positive
2. **If** *count*(Positive SOR Sentences) < *count*(Negative SOR Sentences)
   then $CSO_{(S_1,S_2,...,S_n)}$ = Negative
3. **If** *count*(Positive SOR Sentences) = *count*(Negative SOR Sentences)
   then $CSO_{(S_1,S_2,...,S_n)} = SOR$ of S_k; $IP(S_k) = \max\{IP(S_1), IP(S_2), \ldots, IP(S_n)\}$

Notice that the natural separators for sentences are punctuation marks (period, comma, exclamation sign, question mark, colon, semicolon, etc.), and naturally the sentences would be broken accordingly into sub-sentences at tagging/-parsing time. For complex data inputs, like long paragraphs/sentences or even documents, the expectation is that there would be many sub-sentences participating in multiple compositions. For short paragraphs or snippets, like those exhibited in the Twitter datasets (see Sub-section 6.2.1), we will have to compose semantic orientations for a low number of sub-sentences. By inspecting the data, there would be a composition of 3 to 5 sub-sentences at the most, with the majority of the cases being restricted to 2 or 3 of them. For Twitter data, it is not uncommon that the author of the tweet simply does not use punctuation marks at all, which would result in zero sub-sentences at tokenization time (just one longer than usual sentence).

14.2 Hybrid Advanced Classification (HAC) Method: computing the intensity of polarity

This approach enhances the hybrid standard classification method (HSC) described above by incorporating:

1. *Determination of the intensity of polarity* (IP) with which a given sentence leans towards being *positive* or *negative*.

   The IP of a sentence (X) is to be derived from the IP values of its associated list of sentiment-carrying words $(W_1, \ldots, W_n)$. In other words, the partial IP values of words of a sentence X are to be fused appropriately to derive the global sentence IP value. Mathematically, this problem means that an appropriate mapping $f\colon [0,1]^n \to [0,1]$ is needed to be defined such that:

   $$IP(X) = f(IP(W_1), ..., IP(W_n)).$$

   Fusion operators can be roughly classified into the following categories: conjunctive, disjunctive and compensative:

   (a) *Conjunctive operators* that behave like a logical "and". In this case, the global IP value is high only when all the partial IP values are high but compensation is not possible as the presence of just one small partial IP value will result in a small global value no matter how big the rest of partial IP values are. A well known family of conjunctive operators is the t-norm family, and the minimum operator is the largest of all t-norms.

   (b) *Disjunctive operators* behave like a logical "or", and can be seen as the dual operators of conjunctive operators. In this case, the global IP is low only when all the partial IP values are low. As with conjunctive

operators, compensation is not possible as the presence of just one high partial IP value will result in a high global IP value no matter how low the rest of partial values are. The family of t-conorms belongs to this type of operators, and the maximum is the smallest of all t-conorms.

(c) *Compensative operators* are comprised between the minimum and the maximum, and therefore they are neither conjunctive nor disjunctive. In this kind of operators, a small partial IP value can be compensated by a high partial IP value. This type of operator is also known as an averaging operator with mean, weighted mean and ordered weighted averaging (OWA) operator being widely used in multi-criteria decision making problems.

A class of fusing operators that behaves like a conjunctive operator when all values are low, like a disjunctive operator when all values are high, and like a compensatory operator otherwise, does exist, and it is the family of uninorms operators [255].

Definition 14.1. *A uninorm operator U is a mapping $U: [0,1]^2 \longrightarrow [0,1]$ having the following properties:*

(a) Commutativity: $U(x,y) = U(y,x)$

(b) Monotonicity: $U(x_1,y_1) \geq U(x_2,y_2)$ *if* $x_1 \geq x_2$ *and* $y_1 \geq y_2$

(c) Associativity: $U(x,U(y,z)) = U(U(x,y),z)$

(d) Identity element: $\exists\, e \in [0,1] :\ \forall\, x \in [0,1],\ U(x,e) = x$

Uninorm operators share with t-norm and t-conorm operators the commutativity, associativity and monotonicity properties. Furthermore, the uninorm operator generalises both the t-norm operator and the t-conorm operators. In general, a uninorm operator has an identity element lying anywhere in the unit interval $[0,1]$; a t-norm operator has 1 as its identity element and therefore it is a uninorm operator with identity element 1; while a t-conorm operator has 0 as its identity element and therefore it is a uninorm operator with identity element 0. It is well known that a uninorm operator with identity element $e \in [0,1]$ behaves like (i) a t-norm operator when all partial IP values are below e; (ii) a t-conorm operator when all partial IP values are above e; (iii) a compensative operator in the presence of partial values below and above e. An interesting particular case of uninorm operators are the symmetric aggregative operators, i.e. uninorm operators that have a representation in terms of a single variable function. In particular, the representable uninorm operator with identity element $e = 0.5$ has been characterised as the most appropriate for modelling cardinal consistency of reciprocal preference relations [55].

Based on the above, a general approach in this step would be the implementation of a uninorm operator to derive the IP of a sentence X from the IP of its associated list of sentiment-carrying words. The experimental section reported in Sub-section 14.4.4.2 made use of the minimum operator, which as mentioned above is a type of uninorm:

$$IP(X) = \min\{IP(W_1)\ \ldots\ IP(W_n)\}. \tag{14.1}$$

Once a sentence IP value is obtained, the linguistic labels (granules) $l \in G$ with highest $\mu_l(IP(X))$ is assigned to classify the positive/negative polarity. In those cases when there exist two consecutive labels with equal $\mu_l(IP(X))$, we classify the polarity of the sentence with the label meaning higher as per the ordinal ordering implicitly expressed in the representation given in Fig. 14.5. For example, when $IP(X) = 0.3$, the polarity will be assigned the label *Moderate* rather than the label *Slight*.

2. Diagnosing when a given sentence could be considered rather *objective/neutral* as opposed to either *positive* or *negative*: not all sentences have been created equal, and even in the test dataset that have been carefully chosen, there are some sentences that one could argue that are rather neutral (not leaning towards negative or positive). With the Hybrid Advanced Classification (HAC) system we could consider those sentences classified as having an IP belonging in the *poor interval*, as prime candidates to have a Semantic Orientation that would be leaning more towards *Objective* than to *Subjective*.

14.3 Sentiment lexicon enrichment

This section addresses the classification issue that arises when dealing with sentences for which the data in the lexicon in not enough and a SOR score cannot be produced. A response to such issues is presented in the form of an almost automated approach to enriching the sentiment lexicon by incorporating new opinion-carrying particles into the lexicon to minimise the number of cases when polarity classification is not possible.

14.3.1 Dealing with sentences when the data in the lexicon in not enough

One problem that we will encounter is that a given sentence being processed would include words that are not in the lexicon (none of them are in the lexicon). In such a case, our method cannot provide a *S OR* recommendation. If at least one word would have been included in the lexicon, the *S OR* would have been calculated using the information available, utilising the computation process given in the previous subsection. At this point, the only way forward is to incorporate the new words in the lexicon using some specific criteria as it is shown in the following paragraph. In the interim, no classification can be offered, but once the lexicon is updated, a polarity classification is very possible. Considering the frequency at which SentiWordNet is updated by the on-line feedback provided by regular users (`http://sentiwordnet.isti.cnr.it`), it is fair to say that the growth of the capabilities of our lexicon is guaranteed by the increase on the size of the corpora incorporated into SentiWordNet.

14.3.2 Enriching the sentiment/opinion lexicon

Methodologies based on the utilisation of a sentiment lexicon would eventually come across situations where words carrying opinions in a given sentence are *not* included in the lexicon. The only solution to this problem is to enrich the sentiment lexicon by either adding new words that were not originally part of the aforementioned lexicon or completing/-modifying data attributes already in the lexicon. In this section we will focus on the former case and will explain how to incorporate new opinion-carrying particles into the lexicon.

In order to keep a strict control, we perform off-line the process of adding words to the lexicon using a semi-automated mechanism with some human intervention. The SentiWordNet database can be downloaded and utilised for off-line processing, which is the chosen mechanism. The process we have implemented is given below (every step in the list below uses the output of the previous step) with the pseudo-code required given in Algorithms 15.1, 15.2 and 15.3:

1. Process sentence using NLP techniques (tokenization, negation-handling, PoS tagging, parsing, etc.).
2. Compare sentiment-conveying words found in previous step against sentiment lexicon.
3. Obtain list of the words that were not found in the current/most-recent sentiment lexicon. These words are *candidates* to be added to the lexicon, but not all not-found words will be added to the lexicon; i.e. there are words that are not considered to be sentiment-carrying words, hence they should not be added to the lexicon.
4. Eliminate repeated words and check the PoS group the words belong to.
5. Compare off-line the list of words obtained in previous step against the available SentiWordNet's database and generate list of matches and no-matches.
6. Remove from the list generated in the previous step any particle not-found (there will be words that are not in SentiWordNet). The list of words generated in Step 5 represents candidate-words to be included in our Lexicon. However, those words that are not available in SentiWordNet cannot be added automatically to our lexicon, hence, they are rejected by being removed from the list of matches, and are placed instead in the list of exceptions that will require human intervention in order to make an educated decision.
7. Transform the list of matches from the previous step into a format that enables them to be potentially incorporated into our sentiment/opinion lexicon (the format of the lexicon of our proposed method).
8. Invoke the Lexicon Editor Program (written by our team) to provide a visual interface to an expert to analyse the candidate words already in opinion lexicon format. The human operator will decide the polarity label for each entry as the system prompts her with an input query. The expert (ideally, a linguist) will decide whether the candidate word should be (a) deleted, (b) classified as neutral polarity, or (c) classified as having negative or positive polarity.

9. Add the list of words previously obtained to the existing Sentiment Lexicon in order to generate an *updated* version of the lexicon.

Notice that all steps described above are automated by software that we have developed as part of our prototype, with the exception of step number 8. At a given point in time, some human interaction is usually required to make decisions about whether a given word should be part of the lexicon. A possible partial solution would be to simply add all words found in *SentiWordNet* as the latter is supposed to contain words labelled already as Positive, Negative or Objective. We have taken the latter approach in order to minimise user intervention. However, we still do a visual inspection *before* new words are added to the lexicon, in order to avoid the introduction of noise.

```
Procedure AddNewWords
Data: OldLex, NewLex, NFW are linked-lists
Result: NewLex
while NFW is not empty do
    if NFW.info ∈ SentiWordNet then
        Extract Polarity Scores and PoS particles for words ∈ NFW;
        NewLex ← NewLex ∪ BuildNewLexEntryWith(NFW.info = SentiWordNet.info)
    else
        NewLex ← NewLex ∪ LookUpInDict(NFW.info)
    end
end
return NewLex
```

Algorithm 14.1: Add new words to Sentiment Lexicon

```
Procedure BuildNewLexEntryWith
Data: SentiWordNet all synsets
Result: NewEntryInLexiconFormat
NewEntryLexiconFormat ← Create entry in lexicon format using the data brought in from SentiWordNet return
 NewEntryInLexiconFormat
```

Algorithm 14.2: Generate new entry in lexicon format

```
Procedure LookUpInDict
Data: word, MyDict: word is a term not found in the dictionary and MyDict is our dictionary
Result: NewEntryInLexiconFormat
while MyDict is not empty do
    if MyDict.info = word then
        NewEntryInLexiconFormat ← Obtain polarity of word from dictionary MyDict, and create
          Lexicon-format entry
    else
        NewEntryLexiconFormat ← Create entry in Lexicon format with label indicating: Requires
          manual intervention;
        Report to Linguist expert and generate manual entry into the Lexicon
    end
end
return NewLex
```

Algorithm 14.3: Look Up in Dictionary

14.4 Experimental Results

In this section we will look at the experimental results, starting with the outcome obtained when using the two mentioned SML methods, Naïve Bayes and Maximum Entropy. After this, we will show the results obtained from applying the proposed hybrid method. We will close the section with a comparison of the results. For the preparation and processing of the sentences using the SML classifiers mentioned above, we used extensively the material presented by Bird et al. [26] and Perkins [173].

14.4.1 Experimental Methodology - Summary

As described in Chapter 6, we have followed a rigorous experimental methodology. In brief, the main idea was to take the designated datasets and process them using three different classification methods:

- Naïve Bayes Classifier
- Maximum Entropy Classifier
- The proposed HSC/HAC Classifier

Then, the results of the three aforementioned classifiers would be compared utilising the performance indices described in Section 6.3, with emphasis on *Precision* as that indicator has been utilised previously to establish the current state-of-the-art.

14.4.2 Naïve Bayes classifier

In discussing the Naïve Bayes (NB) classifier, Pang et al. in [165] elaborate that one possible approach to text classification is to assign to a given document d the class $c^* = arg\,max_c\, P(c\,|\,d)$. In order to estimate $P(d\,|\,c)$ (see Chapter 9.1 for all equations and formulae), NB assumes the class features are conditionally independent, with $n_i(d)$ representing the number of possible classification classes (it would be 2 for a binary classifier).

In essence, NB is a probabilistic classifier that is based on the Bayes' theorem. Basically, in the presence of a sample input NB should be able to predict a probability distribution over a set of classes. In this case, word frequencies are the characteristics used to decide whether a paragraph belongs to one category or another. For that to happen, we would have to count with a dictionary (or corpus) previously labelled with the semantic orientation of words (i.e. 'fabulous' conveys a positive intention whilst 'terrible' would convey a bad one). However, despite its apparent simplicity, NB has proven to be very successful in many situations [165].

In the experiments, the NB classifier was trained using some of the recommendations presented by Perkins in [173]. The classifier uses the concept of 'bag of words' [231] to create 'feature vectors' exhibiting the main traits of each sentence. In this case, the NB classifier is a binary classifier, with a sentence being classified either as 'negative' or 'positive', with both categories being exclusive. The *movie_reviews* corpus available with NLTK 2.x was used to train the classifier. The training dataset consists of 1,500 instances (1,500 files containing full paragraphs) that have been pre-labelled as either *positive* or *negative*. There were 500 instances used to test the classifier. Once the classifier has been trained and tested, a *different* dataset of 'movie_reviews' sentences (5,331 sentences pre-labelled as Positive and 5,331 pre-labelled as Negative), was used to evaluate all classifiers (see Sub-section 6.2.2).

The NB classification algorithm returns a probability value that represents the sentence belonging with a specific label (negative or positive). Thus, the probability value has to be 0.5 or higher for a sentence to belong in a specific category (*positive* or *negative*, depending on which case is being tested). With regard to the Twitter Dataset, the classifier was trained using the same data and process described above, until the classifier was ready to be exposed to the test dataset. It is important to notice that the main objective of the use of the Twitter dataset is twofold: (a) to validate the results obtained using the movie database data, and (b) to gain more understanding on the effect of showing shorter sentences -as those typically available in Twitter- to the classifiers. Table 14.5 shows the results obtained after running the NB classifier with the test datasets:

Metric	Twitter A dataset	Movie database dataset
Accuracy	0.6785	0.6717
Precision	0.6315	0.6274
Recall	0.8619	0.8456
F1-Score	0.7315	0.7204

Table 14.5. Naïve Bayes classifier performance indexes

14.4.3 Maximum Entropy classifier

The Maximum Entropy (ME) classification algorithm has been extensively used by the Machine Learning community to deal with text classification problems. One of the main properties of ME is that the algorithm does not make any assumptions about the relationships between features to derive the estimate of $P(c|d)$, which is expressed by the equations and formulae described in Section 9.2. The $\lambda_{i,c}$'s are feature-weight parameters, and a large value for $\lambda_{i,c}$ would imply that "f_i is considered a strong indicator for class c. The parameter values are set so as to maximise the entropy of the induced distribution subject to the constraint that the expected values of the feature/class functions with respect to the model are equal to their expected value with respect to the training data" [165]. Additional information about ME can be found in the literature, for example, in [8, 27, 142].

Following Perkins's recommendations [173], the Generalized Iterative Scaling (GIS) learning method was used to train the ME classifier. As the NB classifier, the ME classifier returns a probability value of the sentence belonging with a specific label (negative or positive), and a probability value equal or higher than 0.5 is required for a sentence to belong in a specific category. Table 14.6 presents the results obtained when the trained classifier is applied to the test datasets:

Metric	Twitter A dataset	Movie database dataset
Accuracy	0.6759	0.6757
Precision	0.6293	0.6291
Recall	0.8610	0.8561
F1-Score	0.7313	0.7253

Table 14.6. Maximum Entropy classifier performance indexes

14.4.4 Proposed hybrid method (HSC/HAC)

The proposed hybrid method has been applied to the test datasets in two different sub-methods or incarnations that grow each time in complexity (HSC and HAC). The results of having applied the classification method HSC (polarity determination) to the test datasets, are followed by the results obtained by applying the HAC method to the test dataset (determining polarity intensity). The proposed hybrid method is first applied to both twitter datasets (see Sub-section 6.2.1), Twitter A dataset and Twitter B dataset, and later on to the movie database dataset (see Sub-section 6.2.2).

14.4.4.1 HSC results

Notice that we have a first and a second pass of the proposed HSC method. When running the experiments, we reset our lexicon to its initial state every time, for the sake of comparison against different data sets. The 2^{nd} pass of the method corresponds to the phase in which the sentiment lexicon and dictionary have learnt new words/terms. The latter mimics better real life scenarios, as every new run of our method should benefit from what it has already learnt.

Table 14.7 presents the HSC (1^{st} pass and 2^{nd} pass) results when applied to the data marked in this article as Twitter A dataset (see Sub-section 6.2.1), while for performance confirmation purposes Table 14.8 presents the HSC 2^{nd} pass results when applied to the data marked in this chapter as Twitter B dataset (see Sub-section 6.2.1). Table 14.9 presents only the HSC 2^{nd} pass results when applied to the test data marked as Movie Review dataset (see Sub-section 6.2.2).

At the beginning of this Chapter we mentioned that we are focusing on Sentiment Analysis at the sentence level. Initial experimental results show that the closer the data utilised reflect the concept of a snippet (a short sentence usually

Metric	HSC (1^{st} pass)	HSC (2^{nd} pass)
Accuracy	0.8273	0.8802
Precision	0.8093	0.8424
Recall	0.8626	0.9451
F1-Score	0.8351	0.8866

Table 14.7. HSC classifier - Twitter A dataset performance indexes

Metric	HSC (2^{nd} pass)
Accuracy	0.8655
Precision	0.8406
Recall	0.8531
F1-Score	0.8332

Table 14.8. HSC classifier - Twitter B dataset performance indexes

Metric	HSC (2^{nd} pass)
Accuracy	0.7585
Precision	0.7278
Recall	0.8257
F1-Score	0.7737

Table 14.9. HSC classifier - Movie Review dataset performance indexes

found in one line or equivalent in social media systems like Twitter), the better our proposed system performs. We will discuss this further in section 14.4.5.

14.4.4.2 HAC results

In subsection 14.4.4.1 we have seen the results obtained by the proposed hybrid method in terms of estimating the polarity of three different datasets. With HAC, we incorporate the fuzzy sets approach already described and as a consequence we can incorporate a *finer granularity level* into the polarity classification process. Details of the results achieved with the Movie Review dataset are provided in Table 14.10 and Table 14.11.

False Negatives	929
No Semantic Orientation (NOSOR)	35
NOSOR (2^{nd} run)	0
True Positives	4,402
Poorly	577
Slight	1,106
Moderate	1,041
Very	1,365
Most	313
Total Number of Sentences	5,331

Table 14.10. HAC classifier increased granularity for **Positive Polarity** dataset

Notice that in the 2^{nd} pass there are no cases of NOSORs. During the 1^{st} pass, the proposed system either learnt new terms (words) that were added to the lexicon, or was capable of finding polarity scores to terms already resident in the sentiment lexicon. With this added granularity to the polarity classification, we can inspect sentences classified in the lower end of the spectrum $[0, 1]$ as well, i.e. sentences labelled as 'poor', and revise them because in terms of classification they could be borderline with Neutral/Objective. Representative examples are sentences like '*The theatre was completed full.*' and "*The Sinner counted with great actors.*", which seem to express facts instead of opinions.

There were no annotations for polarity intensities in any of the utilised datasets. This was expected as the datasets

False Positives	1,646
No Semantic Orientation (NOSOR)	76
NOSOR (2^{nd} run)	0
True Negatives	3,685
Poorly	770
Slight	1,089
Moderate	789
Very	864
Most	173
Total Number of Sentences	5,331

Table 14.11. HAC classifier increased granularity for **Negative Polarity** dataset

were annotated only for polarity (negative or positive). We did annotate 10% of all sentences in the larger dataset though (movie review) in the positive-polarity dataset, which represents approximately 530 sentences distributed as follows: 100 each for the Poor, Slight, Very and Most labels, and 130 for the Moderate one. Table 14.12 presents the indicators for the 10% sample, which are considered very hopeful as efficiency in predicting polarity intensity accurately was above 80% in all cases.

Metric	**Poor**	**Slight**	**Moderate**	**Very**	**Most**
No. of Sentences	100	100	130	100	100
Estimated Correct (%)	81.00	89.00	93.08	91.00	87.00
Estimate Incorrect (%)	19.00	11.00	6.92	9.00	13.00

Table 14.12. Movie Review Positive Polarity dataset sample - HAC classifier performance

14.4.5 Comparison of experimental results

In this subsection we will take a closer look at the experimental results. The first comparison table (Table 14.13) corresponds to results achieved when the different methods were applied to the dataset identified in this article as Twitter A dataset. As stated before, the 2^{nd} pass represents better real life scenarios, as our lexicon has already learnt new terms and their associated properties. The results obtained are very encouraging as the proposed hybrid method improves the results obtained by NB/ME by a significant magnitude. As a performance confirmation exercise for the proposed hybrid method, the data marked as the Twitter B dataset was used. Similar results to those achieved for Twitter A dataset were obtained, with an Accuracy of 0.8655 and a Precision of 0.8406. From now on, our comparison will focus on Accuracy and *Precision* (specially the latter) as the *state of the art* cases we are using for this exercise are based on the principle of using *Precision* as the main element of comparison.

Metric	**NB**	**ME**	**HSC (2^{nd} pass)**
Accuracy	0.6785	0.6759	0.8802
Precision	0.6315	0.6293	0.8424

Table 14.13. Twitter A dataset performance indexes comparison - NB/ME vs. HSC

Moving on to the results obtained using the Movie Review dataset (Table 14.14), we observe that the proposed HSC method performs better than NB/ME, although it is worth mentioning that the overall performance of HSC (for precision) is reduced by approximately 11.46% with respect to its performance on the Twitter datasets. The explanation that we offer for this behaviour is data related. The sentences in the Movie Review dataset are rather complex (a short paragraph or a long sentence made up of a few sub-sentences on average). The type of sentences available in both Twitter datasets are of a simpler nature and they are closer to the concept of a 'snippet'. Let us keep in mind as well that the focus of our research was directed toward presenting a sentiment analysis approach *at the sentence level*. In the following paragraphs, examples of the type of sentences found in the different datasets are provided to support this analysis.

Metric	NB	ME	HSC (2^{nd} pass)
Accuracy	0.6717	0.6757	0.7585
Precision	0.6274	0.6291	0.7278

Table 14.14. Movie Review dataset performance indexes comparison - NB/ME vs. HSC

Movie Review dataset examples

Example 2. *"it was with great anticipation that i sat down to view braveheart last week as it premiered on american cable. the academy award winning film had been highly acclaimed. it also featured the music of one of my favorite film composers , james horner . what i was in for was a disappointing and overlong film which was anything but the best picture of 1995 ..."*

Example 3. *"Vampire's is a rude , chauvinistic movie where women are portrayed as pawns of abuse , present only to pleasure men , feed vampires , readied to be bashed or beaten - till one's sensibilities is shocked by the low iq and mentality of this regressive movie . to make matters worse , the buffoons that go hunting vampires are all rednecks , and deserve to have their heads bitten off , if not , their bodies carved in half ."*

Twitter A dataset examples

Example 4. *"To hell with this economy. I hate aig and their non loan given asses."*

Example 5. *'US planning to resume the military tribunals at Guantanamo Bay ... only this time those DTS on trial will be AIG execs and Chrysler debt holders.'*

Twitter B dataset examples

Example 6. *"There are huge lines at the Apple store."*

Example 7. *"I had to wait for six friggin' hours in line at the Microsoft store. that's not cool man."*

14.4.5.1 Impact of different techniques in hybrid approach

Next we would like to show the improvements in the performance of the proposed system as some of the techniques mentioned in this Chapter were introduced. Table 14.15 shows the precision of the proposed HSC method as specific enhancements were applied one after another, and as such provides a picture of the impact that the different techniques generated as they were added to the proposed solution. Indeed, every step shown in the table inherits the benefits of having introduced a specific technique in the previous step. This results in the precision of the final process being close to 10% higher than that at the start.

Technique incorporated	Precision (%)	Accumulated Impact (%)
Using pre-existing semantic rules	76.77	
Adding effective PoS tagging	79.33	3.33
Adding smart negation handling	81.17	5.73
Adding new semantic rules (R14 & R15)	83.36	8.58
After 2^{nd} pass (once the lexicon has learnt new terms)	84.24	9.73

Table 14.15. Impact of different techniques in hybrid approach precision (Twitter A dataset)

14.4.5.2 Analysis of specific examples

Let us take a closer look at some examples that we believe are of interest. We will start by showing instances that exemplified the different intensities in polarity and their associated linguistic labels, as produced by the proposed HAC method, and then we will continue sharing examples of sentences that have proven to be too hard to classify for the proposed classifier.

14.4.5.2.1 Examples of polarity intensity graduality as per the five linguistic labels introduced

Example 8. *Poor: "effective but too-tepid biopic."*

Example 9. *Slight: "if you sometimes like to go to the movies to have fun, wasabi is a good place to start."*

Example 10. *Moderate: "occasionally melodramatic , it's also extremely effective."*

Example 11. *Very: "the movie's ripe , enrapturing beauty will tempt those willing to probe its inscrutable mysteries ."*

Example 12. *Most: "one of the greatest family-oriented, fantasy-adventure movies ever."*

14.4.5.2.2 Examples of challenging sentences for the proposed hybrid classifier

Example 13. *"spiderman rocks."*
In this case, the proposed classifier does not understand what the term 'rocks' means. As such, the sentence was wrongly classified as having a negative polarity.

Example 14. *"it extends the writings of jean genet and john rechy, the films of fassbinder, perhaps even the nocturnal works of goya."*
This sentence offers the names of great world-class film directors and actors, and claims that the director of the reviewed movie extends their work. However, the lack of context impacts on the ability of the proposed algorithm to classify this sentence properly.

Example 15. *"after watching the movie I found myself between a rock and a hard place."*
In this instance, the use of idioms creates problems for the proposed classifier.

14.4.6 Performance comparison against Machine Learning and state of the art

An accurate and strict comparison cannot be performed unless every method involved is evaluated against exactly the same dataset, and in the case of lexicon driven methods when the same lexicon is used. Therefore, the values shown below are rather *informative* of the performance achieved by each method in different experimental settings. A comparison against state of the art techniques that are not purely machine learning based was not part of this research, but it will be performed in the near future once we execute some of the recommendations offered in Section 14.5 such as the possible replacement of SentiWordNet with SenticNet [45].

Poria et al. [176] provides results for machine learning experiments that we have reused in the comparison shown below in Table 14.16. Notice that the proposed hybrid method does approximately 17% better than the machine learning techniques in general, and 21.50% better against NB/ME. For some time, Socher et al. [198, 199] have been considered state of the art. At the sentence level the proposed hybrid method performs better than Socher et al. [198] and is close to the performance of Socher et al. [199]. Ensemble classification from Poria et al. [176] has achieved the best performance of all, with a precision of 86.21%.

Algorithm	**Precision (%)**
Naïve Bayes (from section 14.4.2)	62.74
Machine learning [176]	67.35
HSC/HAC - Movie Review dataset	72.78
Socher et al. [198]	80.00
HSC/HAC - Twitter dataset	84.24
Socher et al. [199]	85.40
Ensemble classification [176]	86.21

Table 14.16. Proposed hybrid method against state of the art

14.5 Chapter Summary

In general, our proposed hybrid system works very well at the sentence level with a high level of *accuracy* (88.02%) and *precision* (84.24%) when the method is applied against twitter-like datasets. The fact that our hybrid system significantly improved the results obtained using Naïve Bayes (NB) and Maximum Entropy (ME), satisfies our initial hypothesis that a hybrid method using sentiment lexicons, NLP essential techniques and fuzzy sets, should be able to perform well. Another benefit of our proposed system is that we have managed to identify different strengths in the polarity degree of the input sentences with regard to the specific base-case (negative or positive). There is an interesting and intended effect of the introduction of the fuzzy sets component of our method. Those sentences classified in the 'poor' side of the polarity intensity spectrum are prime candidates to be considered rather neutral or objective sentences, instead of subjective (this functionality could be built into a subjectivity determination schema). Also, it allows for a *computing with sentiments* methodology to be realised.

Summarising, in this chapter we have fully described our proposed solution to the sentiment analysis problem at the sentence level. The focus has been on two fundamentals aspects:

- Subjective Polarity Identification (HSC)
- Polarity Intensity Determination (HAC)

In the next chapter (Chapter 15), we will present an extension to our proposed model based on *aggregation by Uninorm*.

Chapter 15

Sentiment Aggregation by Uninorm

> As far as the laws of mathematics refer to reality, they are not certain; and as far as they are certain, they do not refer to reality.
>
> Albert Einstein (1879-1955) [75].

A significant part of the content of this chapter was used in articles published by the author. See references [10, 11, 12, 13, 14, 15, 16, 17, 18]. In this section we will look at a proposed extension to our original model discussed in Chapter 14, which is based on *aggregation by uninorm*. The experimental results associated to this new approach will be presented, too.

15.1 SA Aggregation by Uninorm

There are situations in which lexicon-based methods for Sentiment Analysis (SA) are not able to generate a classification output for specific instances of a dataset. Most often, the reason for this situation is the absence in the sentiment lexicon of terms that are required in the classification effort. In this Chapter we present a method that utilises a *cross-ratio uninorm* to aggregate the classification values produced by two supervised machine learning methods in order to compensate for the lack of ability of our initially proposed hybrid method to offer an immediate classification estimation in situations like the one above described. The solution we are presenting in this Chapter can be seen as an enhancement to the hybrid method for Sentiment Analysis (HAC/HSC) introduced in Chapter 14. The experiments results demonstrate that the newly proposed technique improves the performance of HSC/HAC, as the latter is now capable of always providing an outcome and the outputs obtained by the application of the presented cross-ratio uninorm enjoys very good performance indexes. The study of effective mechanisms for aggregation has been a central part of research in the fields of fuzzy systems and soft-computing [34, 40, 87, 187]. In [187], Rudas, Fodor & Pap mention that "the theory of fuzzy sets today uses a well developed parts of mathematics such as aggregation operations, a generalized theory of relations, generalized measure theory, etc.". Fuzzy sets methods play a key role in many fields, of particular interest for us, in the areas of data fusion, decision-making and group decision-making. In the latter, the clear intention is to combine in a meaningful way the opinion of a number of individuals or methods.

A number of authors have performed in-depth explorations of the utilisation of aggregation functions. A very complete presentation of aggregation and aggregative uninorms can be found in the work of Yager and Rybalov [255] and Rudas and Fodor [186]. In [188] Rudas, Pap & Fodor show how key information fusion is in many complex areas like decision making, utility theory, fuzzy inference systems, robotics and vision. The authors cover aggregation functions and their fundamental properties, with four main classes of aggregation functions being identified. In [107], Jočić and Štajner-Papuga focus on pairs of binary aggregation operators on the unit interval that verify the distributivity law, which is important in the utility theory. More recently, Wu et al. [245] present an interesting discussion on the use of aggregation methods for group decision-making in the specific context of social networks. The authors investigate a uninorm based approach to propagate trust through a network. In [270] the authors propose a new method to apply in group decision-making context with incomplete reciprocal preference relations. The method performs a multiplicative consistency analysis of the opinions of each expert, and provides an aggregation.

In [242, 243, 246], the mathematical modelling of the multiplicative transitivity property originally introduced by Tanino for reciprocal fuzzy preference relations is investigated and derived for the case of intuitionistic reciprocal preference relations. They use as a starting point Zadeh's extension principle and the horizontal representation theorem of fuzzy sets based on the concept of alpha level set. Their findings assist the authors in the building of a novel consistency based induced ordered weighted averaging operator. According to these researchers, the aforementioned operator is capable of associating a higher contribution in the aggregated value to the more consistent information. In [217], Ureña et al. present an approach to decision-making based on intuitionistic preference relations, which provide a simple but flexible representation structure of experts' preference on a set of alternative options, "while at the same time allowing to accommodate degrees of hesitation inherent to all decision making processes." [217]. In addition, the authors introduce the concept of expert's confidence which is based on the hesitancy degree of the reciprocal intuitionistic fuzzy preference relations. Then, they provide a group decision-making procedure, "based on a new aggregation operator that takes into account not only the experts' consistency but also their confidence degree towards the opinion provided.". Meng and Chen address a new method to deal with group decision making with incomplete fuzzy preference information in [146] based on the application of an induced hybrid weighted aggregation operator. A particular feature of this aggregation operator is that "the group consistency is no smaller than the highest individual inconsistency, and the group consensus is no smaller than the smallest consensus between the individual fuzzy preference relations." [146].

As mentioned before, it is not uncommon for lexicon-based sentiment analysis methods to be compromised when processing sentences containing terms/words that are not in the lexicon. In situations like this there are some palliatives that could be applied, some of which are described in Chapter 14, like the use of a previously-generated word dictionary or vocabulary that could assist in finding an alternative solution. In general, the options available when a lexicon-based method cannot deliver an answer are:

- Addition of the missing word(s) into the lexicon, as suggested in Chapter 14. This option is a valid one, provided that there is no noise introduced into the lexicon. In order to guarantee the latter, human intervention may be required, which would prevent the classification system to provide an immediate answer.
- Utilise a word-frequency dictionary, as mentioned before. This, however, may not always produce a good answer and additionally, it is typically expensive from the computational standpoint.
- Introduce a method that is not lexicon-dependent, like a machine learning algorithm like the one presented in Poria et al. [176] or another option, like Naïve Bayes.
- Select a proper aggregation technique that could smartly fusion the classification outputs of two or more algorithms that are not lexicon-dependent. In our particular case, we propose the aggregation of the outcomes of two supervised machine learning techniques: Naïve Bayes (NB) and Maximum Entropy (ME), utilising a cross-ratio uninorm $U(x,y)$.

In this Chapter we have chosen to propose the latter option among those presented above for reasons that will become evident as we progress with the presentation of the material.

In Chapter 14 we proposed a hybrid method to the Sentiment Analysis / Opinion Mining problem at the Sentence Level. By using semantic rules, fuzzy sets, unsupervised machine learning techniques and a sentiment lexicon improved with the support of SentiWordNet [79], the method proved to be better than some of the established *supervised machine learning* techniques when they are used in isolation. One of the challenges of our proposed method was producing a classification outcome when the opinion lexicon utilised did not contained *at least* one sentiment-conveying word of those present in the sentence being processed. In the previous Chapter, we presented the success of our Hybrid Standard Classification (HSC) System and the Hybrid Advanced Classification (HAC) System that work as a two-step approach: (a) determine polarity, and (b) compute the intensity of the previously determined polarity. In this Chapter we propose the introduction of a mechanism to rely on when HSC/HAC cannot issue a classification value. We will resort to aggregating the output of a number of supervised machine learning methods in order to be capable of always producing a classification outcome. The proposed aggregation method is a cross-ratio uninorm. As aforementioned, the set of uninorm operators has both the set of t-norm operators and the set of t-conorm operators as its subsets. Indeed, a uninorm operator with identity element "$e = 1$" becomes a t-norm operator; while a uninorm operator with identity element "$e = 0$" becomes a t-conorm operator.

In general, a uninorm operator with identity element $e \in]0,1[$ behaves like (i) a t-norm operator when all aggregated values are below e; (ii) a t-conorm operator when all aggregated values are above e; (iii) a compensative operator otherwise.

Notice that the semantic orientation discrimination between *positive*, *negative* or *neutral* is in accordance with the behaviour of uninorm operators. Thus, based on the above, we suggest that when a lexicon-based method, like the HSC/HAC technique, is unable to derive the polarity of a sentence then an alternative approach could consist of implementing a uninorm operator to aggregate the polarity classification outputs, $\{x_1, x_2, \ldots, x_n\}$, of non-lexicon dependent classification methods, $\{m_1, m_2, \ldots, m_n\}$, respectively. Thus, the resulting aggregation would be defined by $U(x_1, x_2, \ldots, x_n) = \Lambda$, where $\Lambda \in [0,1]$ and U is an appropriate uninorm operator. In the following section, we present the class of cross-ratio uninorm operators implemented in this alternative approach to SA.

15.2 Cross-ratio Aggregative Uninorm Operators

Neither t-norm operators nor t-conorm operators allow "low" values to be compensated by "high" values or viceversa. However, as explained above "uninorm operators may allow values separated by their identity element to be aggregated in a compensating way" [86].

Yager and Rybalov [255] provided the following representation of uninorms in terms of a strictly increasing continuous function of a single variable $\phi\colon [0,1] \longrightarrow [-\infty, \infty]$ (generator function):

$$U(x,y) = \phi^{-1}\left[\phi(x) + \phi(y)\right] \ \forall x, y \in [0,1]^2 \backslash \{(0,1),(1,0)\}.$$

such that $\phi(0) = -\infty$, $\phi(1) = +\infty$. Chiclana et al. in [55] proved that the and-like representable uninorm operator with $e = 0.5$ and $\phi(x) = \ln \frac{x}{1-x}$ [117], known as the cross-ratio uninorm,

$$U(x,y) = \begin{cases} 0, & (x,y) \in \{(0,1),(1,0)\} \\ \dfrac{xy}{xy + (1-x)(1-y)}, & \text{Otherwise.} \end{cases} \tag{15.1}$$

is the solution to the functional equation modelling the concept of cardinal consistency of reciprocal preference relations. The cross-ratio uninorm operator has also been utilised in the influential PROSPECTOR expert system [20]. Fodor [86] extended the cross-ratio uninorm with the identity element $e = 0.5$, so the identity element e can take on any value in $]0,1[$:

$$U(x,y) = \begin{cases} 0, & (x,y) \in \{(0,1),(1,0)\} \\ \dfrac{(1-e)xy}{(1-e)xy + e(1-x)(1-y)}, & \text{Otherwise.} \end{cases} \tag{15.2}$$

Expression (15.2) presents the cross-ratio uninorm as an aggregation operator of two arguments. However, associativity property allows uninorm operators to fuse n (> 2) arguments:

$$U(x_1, x_2, \ldots, x_n) = \begin{cases} 0, & \text{if } \exists i, j : (x_i, x_j) \in \{(0,1),(1,0)\} \\ \dfrac{(1-e)^{n-1} \prod_{i=1}^{n} x_i}{(1-e)^{n-1} \prod_{i=1}^{n} x_i + e^{n-1} \prod_{i=1}^{n} (1-x_i)}, & \text{Otherwise.} \end{cases} \tag{15.3}$$

Values in the interval $[-1,1]$ could be used as well. Indeed, if we were interested in having semantic orientation values in $[-1,1]$, then according to [186], there is the possibility of using the modified combining function $C\colon [-1,1]^2 \to [-1,1]$ proposed by van Melle [37]:

$$C(x,y) = \begin{cases} x + y(1-x) & ,if \min(x,y) \geq 0 \\ x + y(1+x) & ,if \max(x,y) \leq 0. \\ \dfrac{x+y}{(1 - \min(|x|,|y|)} & ,\text{Otherwise.} \end{cases} \tag{15.4}$$

Notice that C is not defined in the points $(-1,1)$ and $(1,-1)$. However, as per Rudas and Fodor [186], rescaling function C to a binary operator on $[0,1]$, it is possible to obtain a representable uninorm with identity element 0.5 and "as underlying t-norm and t-conorm the product and the probabilistic sum." [186]. This result allows therefore to provide the following definition of C in $(-1,1)$ and $(1,-1)$: $C(-1,1) = C(1,-1) = -1$. In this Chapter we will not be using

equation (15.4), but it has been introduced here in an effort to show the generalisation in the method being proposed if semantic orientation values in $[-1, 1]$ were to be used.

15.3 The Proposed Uniorm-driven aggregation mechanism to consolidate the output of selected supervised machine learning algorithms

Our hybrid method will continue working as initially proposed and expected for polarity and polarity intensity classification, using the same sentiment lexicon, semantic rules and the rest of the elements originally presented in the previous Chapter. The changes we are about to describe are primarily occurring to the outputs representing the semantic orientation previously produced by the two supervised machine learning approaches, NB and ME, and to the new aggregated semantic orientation that our method will produce *in absentia* of any other/better output available.

As we mentioned before, polarity scores and semantic orientation polarities were measured using what we have called the pair $\langle ps, pl \rangle$, where ps = Polarity Score $\in [0, 1]$, with 0 representing an *extremely weak polarity* and 1 representing a *very strong one*, and pl = Polarity label $\in \{Positive, Negative\}$. In order to work in this new complementary approach a conversion must be enforced to the unit interval that accounts for negative and positive polarities represented by a given number $\in [0, 1]$, without having to utilise an additional bit of information to represent the positive/negative label. As discussed in the previous section, we must be capable of differentiating the values that Positive Scores and Negative Scores can assume as well as the resulting semantic orientation of any possible combination among them. As a consequence, we have mapped the already described polarity/semantic scores pair $\langle ps, pl \rangle$ for all outputs of the NB and ME methods, to a new interval as follows:

- Negative values represented by the pair $\langle ps, \text{Negative} \rangle$, where $ps \in [0,\ 1]$ are mapped to $[0,\ 0.4999]$.
- Positive values $\langle ps, \text{Positive} \rangle$, where $ps \in [0,\ 1]$ are mapped to $[0.5001,\ 1]$.

Now, all scores belong in the unit interval, leaving the value of 0.5 as the neutral element e representing Objective/Neutral semantic orientation. This value of $e = 0.5$ corresponds with the identity element introduced in equations (15.1) and (15.2). As a consequence, the new introduced polarity spectrum maps to two symmetrical half unit-interval ranges and their values are ready to be aggregated by a cross-ratio uninorm having as neutral element $e = 0.5$. Full details about the science behind these equations and associated approach can be found in Section 13.2 and Section 15.2 of this document.

15.3.1 The proposed aggregation process (HACACU)

The proposed aggregation process describe in this section is called **H**ybrid **A**dvanced **C**lassification with **A**ggregation by **C**ross-ratio **U**ninorm (HACACU). Let us recall that the outputs of Naïve Bayes (NB) and Maximum Entropy (ME) are *numbers* that belong in the unit interval, and the cut off point is given by values that are either greater than or less than 0.5.

If Output(NB) > 0.5 → Semantic Orientation is *Positive*
If Output(NB) < 0.5 → Semantic Orientation is *Negative*.

If Output(ME) > 0.5 → Semantic Orientation is *Positive*
If Output(ME) < 0.5 → Semantic Orientation is *Negative*.

The above If-Then clauses satisfy our criteria of the neutral element being $e = 0.5$.

In the case that our Hybrid Classification Method is incapable of producing a classification as a consequence of some of the limitations of lexicon-based techniques, then the proposed alternative option is described as:

1. Collect the outputs of NB
2. Collect the outputs of ME

3. For each sentence, feed both outputs -NB and ME- to the cross-ratio uninorm presented above in equation (15.1), which will produce a value (*Result*) for each sentence

   (a) If Result > 0.5 then Semantic Orientation = *Positive*

   (b) If Result < 0.5 then Semantic Orientation = *Negative*

   (c) If Result = 0.5 then Semantic Orientation = *Neutral*

Graphically, the proposed complementary method is presented in Fig. 15.1. Notice that this proposed improvement

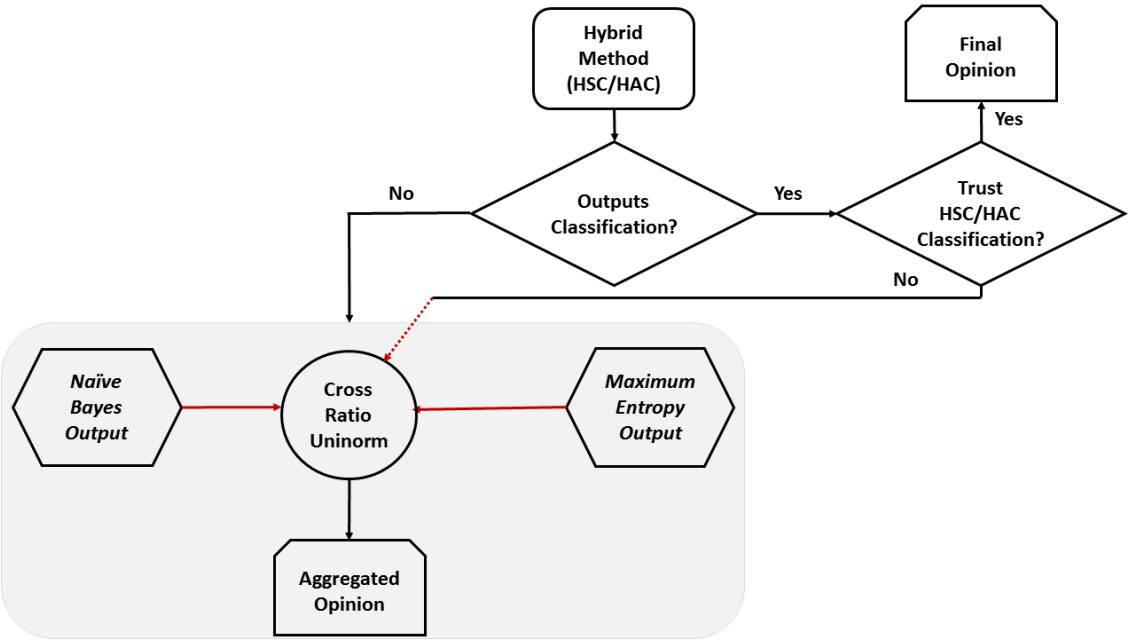


Fig. 15.1. *Enhanced Option for Hybrid Classification Method - Cross-ratio Uninorm Aggregation* (shaded area)

ensures that our enhanced **H**ybrid **A**dvanced **C**lassification Method with **A**ggregation (HACA) -as an extension to the method we presented in Chapter 14- is **always** in a position to produce a classification output.

15.4 Experimental results

In the method proposed in Chapter 14, when the sentiment lexicon did not contain the necessary terms/words to produce a classification output when processing a given sentence, there were only two possible paths to follow: (a) add terms to the lexicon (off-line process), or (b) use the services of a Word-frequency dictionary (on-line process) that is computationally costly to build. The latter method has been utilised in our HSC method [13], as a last resort, when the necessary terms for producing a classification are not present in our sentiment lexicon. Experimental results are reported in this section regarding the performance of the alternative path proposed in this Chapter based on the cross-ratio uninorm operator.

15.4.1 Experimental Methodology - Summary

In terms of running the experiments to test the success of the semantic aggregation by cross-ratio uninorm, the situation was a bit different than the one faced when comparisons were made against supervised machine learning methods that required previous training. In the case of the experiments results presented in this chapter, comparisons are established between the proposed uninorm method, and other techniques that do not require training, namely *arithmetic mean* and *word-frequency dictionary*. The process utilised was to run the specific methods using their own idiosyncratic approaches: (a) calculating the arithmetic mean of the outputs of the NB and ME methods, (b) searching in the word-frequency dictionary for the semantic orientation polarity of previous occurrences of specific sentiment-conveying words, and (c) obtaining the polarity orientation by using the proposed HSC method under different scenarios (addressing the problem

through the word-frequency dictionary, applying the cross-ratio uninorm strategy, or enabling the addition of new terms to the sentiment lexicon).

15.4.2 Datasets utilised

We make use of the Movie Review Dataset provided by Pang and Lee and available at `http://www.cs.cornell.edu/people/pabo/movie-review-data/` . In addition, we utilise a dataset containing Twitter data, *Sentiment140*, which is available at `http://help.sentiment140.com/for-students`.

15.4.3 Results for the application of Cross-ratio Uninorm Aggregation to test datasets

In order to assess the validity of the alternative path based on the cross-ratio uninorm proposed here, experiments were carried out to compare:

1. The cross-ratio uninorm performance as an aggregation tool of the classification outputs when applying the NB and ME algorithms against the performance achieved using the *arithmetic mean* aggregation instead, and the classification performance using exclusively the services of a *word-frequency dictionary* (Subsection 15.4.3.1);
2. The performance of the cross-ratio uninorm implementation in the HSC lexicon-based method against the performances of the HSC lexicon-based method with the off-line addition of missing-words to the lexicon and with the Word-frequency dictionary, respectively, when the lexicon cannot respond (Subsection 15.4.3.2). The performance of the Cross-ratio uninorm method when embedded in our hybrid method (HSC) as a complement

15.4.3.1 Cross-ratio uninorm against two other possible techniques

Tables 15.1 and 15.2 show comparative results of the cross-ratio uninorm aggregation of the classification outputs when applying the NB and ME algorithms against two other alternative methods: (a) the arithmetic mean, and (b) the classification outputs obtained by using the Word-frequency dictionary described in Chapter 14. Experiments have been performed for both the Movie DB dataset (10,662 occurrences, the complete set) and the Twitter dataset (15,000 occurrences).

Alternative Method	**Accuracy**	**Precision**	**Recall**	**F1-score**
Arithmetic Mean	0.46	0.52	0.47	0.49
Word-frequency Dictionary	0.58	0.66	0.57	0.61
Cross-ratio Uninorm (NB & ME)	0.66	0.67	0.64	0.66

Table 15.1. Method Vs. Indicators (Movie DB: 10,662 sentences)

Alternative Method	**Accuracy**	**Precision**	**Recall**	**F1-score**
Arithmetic Mean	0.50	0.57	0.43	0.49
Word-frequency Dictionary	0.56	0.69	0.56	0.62
Cross-ratio Uninorm (NB & ME)	0.75	0.78	0.82	0.80

Table 15.2. Method Vs. Indicators (Twitter dataset: 15,000 sentences)

In this comparison, the cross-ratio uninorm (NB & ME) comes ahead of the other two algorithms in all four performance indicators. Particularly, the recall indicator (how many of the true positives sentences were found) displayed by the cross-ratio uninorm is the highest of all by a significant margin. In the next section, when the sentiment lexicon cannot respond, the cross-ratio uninorm (NB & ME) technique enhancement to our HSC method will be compared against the HSC lexicon-based approach with the off-line addition of missing-words to the lexicon and with the Word-frequency dictionary, respectively.

15.4.3.2 Cross-ratio uninorm as an enhancer of our hybrid method

In the hybrid model we presented in Chapter 14 we did show that during the first pass of the proposed hybrid algorithm, there were sentences that could not be classified as the sentiment lexicon did not count with the required terms/words in

order to produce a classification outcome. Two approaches to circumvent this problem were proposed: (a) the use of a word-frequency dictionary that served its purpose but it has a negative aspect in that its creation involves an algorithm of complexity $O(n^2)$, where n is the numbers of words in the dataset being utilised; or (b) incorporate new terms into the dictionary, which could not be done interactively and required an expert human intervention.

The next step in our experimental phase is to study and analyse the performance of the cross-ratio uninorm (NB & ME) when embedded in our HSC method. To force the application of the cross-ratio uninorm, terms were randomly removed from the sentiment lexicon leading to a number of sentences that could not be classified using the HSC method (1,337 sentences in total). Hence, the services of: (1) the Word-frequency Dictionary; (2) the Cross-ratio Uninorm; and (3) off-line addition of missing terms to the sentiment lexicon, were demanded for those 1,337 sentences. The obtained results are presented in Table 15.3 (for Movie Dataset) and Table 15.4 (for Twitter Dataset).

Method	**Accuracy**	**Precision**	**Recall**	**F1-score**
(1) Hybrid using Word Dictionary	0.66	0.64	0.77	0.69
(2) Hybrid using Cross-ratio Uninorm	0.74	0.71	0.82	0.76
(3) Hybrid with word-addition enabled	0.76	0.73	0.83	0.77

Table 15.3. All Hybrid methods derived from HSC [13] - Movie Dataset

We see that the cross-ratio uninorm approach performs second-best in the group, achieving results that are very close to those attained by HSC with word-addition enabled (only 2% below in Accuracy and Precision). In third place we get the Word-frequency Dictionary that is 8% and 7% worse than the cross-ratio uninorm for Accuracy and Precision, respectively.

We repeated the same experiment using this time the Twitter Dataset, obtaining the results presented in Table 15.4, which basically confirms the previous results and analysis. We can say with confidence that the alternative option when the lexicon cannot offer a solution based on the Cross-ratio Uninorm constitutes an excellent alternative at a very low cost, both computationally and people-wise.

Method	**Accuracy**	**Precision**	**Recall**	**F1-score**
(1) Hybrid using Word Dictionary	0.77	0.74	0.84	0.79
(2) Hybrid using Cross-ratio Uninorm	0.86	0.81	0.93	0.87
(3) Hybrid with word-addition enabled	0.88	0.84	0.94	0.89

Table 15.4. All Hybrid methods derived from HSC [13] - Twitter Dataset

15.5 Chapter Summary

By using the cross-ration uninorm for aggregating the outcomes of NB and ME algorithms when the sentiment lexicon cannot assist in producing a classification, we see a slight improvements in *precision* when compared to the results achieved by recurring to the word-frequency dictionary. The magnitude of this precision betterment is of approximately 0.0395 (3.95%) for the movie database full dataset and of 0.0055 (0.55%) for the complete twitter dataset. Further conclusions and future research regarding the sentiment aggregation by uninorm introduced in this chapter will be provided in Chapter 18.

Summarising, in this chapter we have shared a proposed enhancement to our hybrid model addressing the SA problem which is based on *aggregation by Uninorm*. In the next chapter (Chapter 16) we will discuss another model that could improve our hybrid method, but this time the technique is based on *aggregation by Consensus*.

Chapter 16

Sentiment Aggregation by Consensus

> "Reason cannot defeat emotion, an emotion can only be displaced or overcome by a stronger emotion.".
>
> Baruch Spinoza [1632-1677]

A significant part of the content of this chapter was used in articles published by the author. See references [10, 11, 12, 13, 14, 15, 16, 17, 18]. In this section we will look at a proposed extension to our original model discussed in Chapter 14, which is based on *aggregation by consensus*. The experimental results associated to this aforementioned approach will be presented, too.

16.1 Consensus in SA

In group decision-making there are many situations where the opinion of the majority of participants is critical. The scenarios could be multiple, like a number of doctors finding commonality on the diagnose of an illness or parliament members looking for consensus on an specific law being passed. In this Chapter we present a method that utilises Induced Ordered Weighted Averaging (IOWA) operators to aggregate a majority opinion from a number of Sentiment Analysis (SA) classification systems, where the latter occupy the role usually taken by human decision-makers as typically seen in group decision situations. In this case, the numerical outputs of different SA classification methods are used as input to a specific IOWA operator that is semantically close to the fuzzy linguistic quantifier 'most of'. The object of the aggregation will be the intensity of the previously determined sentence polarity in such a way that the results represents what the majority stand for. During the experimental phase, the use of the IOWA operator coupled with the linguistic quantifier 'most of' (IOWA_{most}) proved to yield superior results than those achieved when utilising other techniques commonly applied when some sort of averaging is needed, such as arithmetic mean or median techniques.

16.2 Discussion

Group decision making is a task where a number of agents get involved in a decision process to generate a value that represents their individual decisions in the group process [97]. In the case of the Sentiment Analysis (SA) problem research effort presented in this Chapter, the agents would be any number n of SA classification methods, where $n \geq 2$. Experiments have been conducted using three methods: (a) Naïve Bayes [165], (b) Maximum Entropy [27], and our (c) Hybrid Approach to SA problem reported in Chapter 14.

Additional insights are provided on how to continue improving the results obtained in our Hybrid Approach to the SA Problem mentioned before. The central idea in this Chapter is that several classification methods could be utilised in such a way that each of them performs the classification task following their intrinsic characteristics and design principles. Then, a proper model should be put in place to account in a sensible manner for the opinions of all of them. We would like to obtain a classification value that articulates all of them, that summarises the collective opinion of these methods, with the caveat that we would like the final classification value *to reflect the opinion of the majority*. In particular, we would like to compute what the opinion of the majority is with respect to the intensity of the polarity of a given sentence.

The concept of aggregating diverse methods recommendations is not technique dependent. Nevertheless, a proper aggregation method must be chosen. *Arithmetic mean* and *median* are central tendency values/techniques that have been used in the past [173]. However, we are in search of an aggregation mechanism that gives more importance to the classification output of some methods depending on the characteristics of the values they produce. As a consequence, we propose the use of an IOWA operator [54, 258] to aggregate the outcome of several opinion classification methods using an induced guiding principle. We will discuss further Yager's Ordered Weighted Averaging (OWA) operator [256] and will present our rationale for having selected one of them as the aggregation mechanism of choice. In order to test our ideas, we will utilise the classification outcomes of Naïve Bayes, Maximum Entropy and the Hybrid Advanced Classification method presented in Chapter 14 as the individual polarity classification values to derive a consensus IOWA majority based polarity classification for the SA problem.

16.3 Consensus aggregation - Related work

A number of authors have explored the utilisation of members of the family of OWA operators in different situations and domains, with Pasi and Yager providing comprehensive information about OWA operators [168]. Boroushaki and Malczewski [33] present a very interesting example of applying a fuzzy majority approach for Geographical Information Systems (GIS) based on multi-criteria group decision-making. Bordogna and Sterlacchini [31] address a multi criteria group decision making process based on the soft fusion of coherent evaluations of spatial alternatives (GIS-Spatial Analysis). In the work of Wei and Yuan [227], we can learn about the application to coal mine safety of linguistic aggregation operators in order to achieve effective decision-making. Mata et al. [144] presents the utilisation of a Type-1 OWA operator [271] as a vehicle to obtain aggregation in the presence of unbalanced fuzzy linguistic information in decision making problems, while Chiclana and Zhou [52] demonstrate that type-2 fuzzy sets can be effectively defuzzified using a Type-1 OWA alpha-level aggregation approach [272].

As we know, in multiple attribute decision-making situations, optimistic and pessimistic extremes are represented by maximum and minimum. Wei et al. [228] propose a method based on Induced OWA operators in multiple attribute decision-making, in order to capture human attitudes that fall between the two extremes points of optimism and pessimism. In [180], Qian and Xu extend the properties of IOWA operators by incorporating linguistic preference information in applications in group decision making. Mata et al. [143] propose a Type-1 OWA methodology devised to achieve consensus in multi-granular linguistic contexts. The work of Peláez et al. [169] on OWA operators in decision-making aimed to obtaining the opinion of the majority is very influential. More recently, Yager and Alajlan [253] addressed again the problem of obtaining a consensus subjective probability distribution from the individual opinions of a group of agents about the subjective probability distribution. In [254], Yager and Alajlan revise the parameterization aspects of OWA aggregation operators. The authors stress the fact that the aforementioned parameterization is achieved by the characterizing OWA weights. Yager and Alajlan expand on a number of different paths to provide these characterizing OWA weights. As typically the importance of the values being aggregated is application-dependent and the arguments have different importances, it becomes key "appropriately combining the individual argument weights with the characterizing weights of the operator to obtain operational weights to be used in the actual aggregation" [254]. The authors present "the use of a vector containing the prescribed weights and the use of a function called the weight generating function from which the characterizing can be extracted" [254]. Finally, it is worth mentioning Perkins's work [173] on the use of median, voting and arithmetic mean when aggregating multiple classification results as relevant to the present work.

16.4 Fuzzy Majority in Collective Decision Making modelled with an IOWA Operator

It has been already established by Yager [250, 258] that the OWA operator provides a *parameterized* family of mean type aggregation operators. The parameterized aspect is directly associated to the weighting vector. In this section we will take a closer look at OWA operators, fuzzy majority and other related decision making aspects.

16.4.1 The Linguistic Quantifier in Fuzzy Logic

The same way as other fuzzy logic concepts relate to classical logic, the linguistic quantifier generalises the idea of quantification of classical logic. In classical logic there exist two types of quantifiers that can be used in propositions: the universal quantifier (for all) and the existential quantifier (there exists). According to Pasi and Yager [168], by using linguistic quantifiers we are capable of referencing a variable number of elements of the domain of discourse. This referencing can be done in a *crisp* way or in a *vague* (fuzzy) manner. See Table 16.1 for more details. Pasi and Yager

Referencing type	**Examples**
Crisp	*at least k* of the elements, *half* of the elements, *all* of the elements
Vague (fuzzy)	*most* of the elements, *some* of the elements, *approximately k* of the elements

Table 16.1. Crisp and fuzzy referencing to elements of the domain of discourse

[168] differentiate between two types of fuzzy quantified propositions as presented in Table 16.2. According to Zadeh

Type of fuzzy quantified proposition	**Components**	**Statement**	**Examples**
Q X are Y	Q = Linguistic quantifier, Y = a fuzzy predicate, X= set of elements	Q elements of set X satisfy the fuzzy predicate Y	Most of the criteria are satisfied by alternative A_i, in which Q = *most*, X = the set of the *criteria*, and Y = *satisfies alternative* A_i
Q B X are Y	Q = Linguistic quantifier, B & Y = a fuzzy predicates, X= set of elements	Q elements of set X which satisfy the fuzzy predicate B also satisfy the fuzzy predicate Y	Most of the important criteria are satisfied by alternative A_i, in which B = *important*

Table 16.2. Types of fuzzy quantified propositions

[260], in fuzzy logic the quantifiers have been defined as fuzzy subsets of two main types: absolute and proportional. As discussed in [168], "absolute quantifiers, such as *about* 7, *almost 6*, etc. are defined as fuzzy subsets with membership function $\mu_Q : \Re^+ \rightarrow [0,1]$, where $\forall x \in \Re^+$; $\mu_Q(x)$ indicates the degree to which the amount x satisfies the concept Q. In addition, as per [168], proportional quantifiers like *most*, or *about* 70%, are defined as fuzzy subsets of the unit interval: $\mu_Q : [0,1] \rightarrow [0,1]$, where $\forall x \in [0,1]$, $\mu_Q(x)$ indicates the degree to which the proportion x satisfies the concept Q. For sake of simplicity from now on we will indicate μ_Q by Q and $\mu_Q(x)$ by $Q(x)$".

16.4.2 Linguistic Quantifiers as soft specifications of majority-based aggregation

The focus of the discussion in this Chapter will be on monotonic non-decreasing linguistic quantifiers, like *most* and *at least*. We will concentrate on the quantifier "most" as we are attempting to model a *majority*. As per Pasi and Yager [168], our objective is use linguistic quantifiers "in guiding an aggregation process aimed at computing a value which synthesizes the majority of values to be aggregated". The aimed value is known as a 'majority opinion'. Notice that in multi-agent decision making the synthesis or aggregation of a majority opinion is a key topic. As we will discuss later on in this chapter, OWA operators can be built on top of the concept of the fuzzy definition of a linguistic quantifier. In [168], the authors discuss extensively whether the result of aggregating a collection of values with a quantifier that corresponds to the concept of *majority*, will be representative of the majority of values. The semantics behind the aggregation being performed is the key to reflecting the concept of a *majority*. As such, the two alternatives in terms of OWA semantics presented in [168] are: (1) OWA operators as an aggregation guided by 'majority' linguistic quantifiers, and (2) IOWA operators as drivers of a *majority opinion*.

1. Case 1 - The semantic of OWA operators is an aggregation guided by 'majority' linguistic quantifiers [168]: The authors see the OWA operator as an aggregation operator taking a collection of argument values and returning a value. As we know, the weights of the OWA operator will determine the behaviour of the aggregation operator.

   - One possible semantics is the one that presents the OWA operator as a generalisation of the idea of an averaging or summarising operator, i.e. $w_i = 1/n$ for all i yields the simple average as all elements in the aggregation contribute equally to the final result.
   - Another semantics for the OWA operator to be considered, is the one in which the OWA operator is a generalisation of the classical logic quantifiers *there exists* and *for all*.

   In [168], Pasi and Yager claim that these semantics of the aggregation do *not* really reflect the concept of majority in group decision making applications. Hence, the authors suggest the following different approach.

2. Case 2 - Using IOWA operators to obtain a *majority opinion*: this corresponds to the concept of majority as typically used in group decision making applications, where more than one agent participates, and it is closer to the linguistic quantifier *most*. "We want an evaluation that correspond to a majority of the experts holding a similar opinion, where by majority we intend *most*" [168]. In fact we require an operator that calculates an average-like aggregation of "a **majority** of values that are similar".

   Pasi and Yager [168] propose an aggregation that according to them *does have* a majority semantics. That proposal is based on the utilisation of OWA operators "with an inducing ordering variable which is based on a proximity metric over the elements to be aggregated." [168]. The authors focus on a method for calculating the weights used in the OWA operator that would allow them to obtain the weights from a functional form $Q : [0,1] \to [0,1]$ such that $Q(0) = 0$, $Q(1) = 1$, and $Q(x) \geq Q(y)$ for $x > y$ corresponding to a fuzzy set representation of a proportional monotone qualifier. "For a given value $x \in [0,1]$, the $Q(x)$ *is the degree to which x satisfies the fuzzy concept being represented by the quantifier*" [168]. Based on function Q, the OWA vector is determined in the way described below [168]:

$$w_i = Q(i/n) - Q((i-1)/n) \tag{16.1}$$

   Hence, w_i represents the increase of the satisfaction in getting i with respect to $(i-1)$ criteria satisfied. The authors claim that, for example, if they were going to define the so-called weighting vector of the OWA operator that is associated to the linguistic quantifier **most**, a possible *membership function* of the **most** quantifier would be:

$$\mu_{most}(x) = \begin{cases} 1 & \text{if} \quad x \geq 0.9 \\ 2x - 0.8 & \text{if} \quad 0.4 < x < 0.9 \\ 0 & \text{if} \quad x \leq 0.4 \end{cases} \tag{16.2}$$

   The main idea being pursued is that "most similar values must have close positions in the induced ordering in

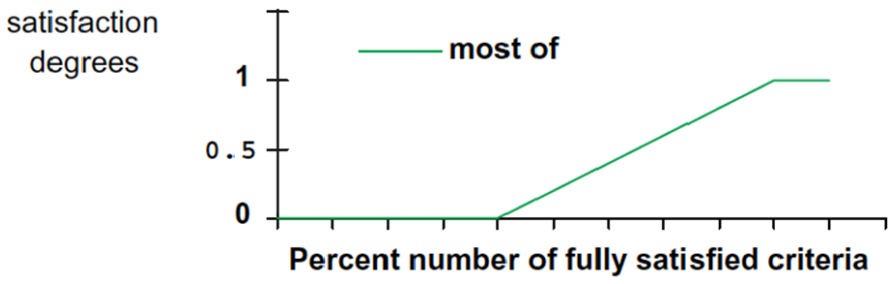


Fig. 16.1. A possible definition of the linguistic quantifier *most*, as presented in [168], page 395

   order to appropriately be aggregated" [168]. We interpret this as similar values should be closer to each other in the support vector and the final output should reflect in a closer manner the opinion of the majority. "To this

aim our intent is to take the most similar values in the quantity specified by the quantifier and apply to them an averaging operator." [168]. What is needed is the ability to calculate the similarities between the opinion values being considered. "The values of the inducing variable of the IOWA operator are obtained by means of a function of the *similarities* between *pairs* of the opinion values." [168]. Such a function is defined using a support function introduced by Yager in [257]. We reproduce below the calculation presented by Yager, in order to obtain the aforementioned support function.

A support function, Sup [257], is a binary function that calculates a value $Sup(a,b)$ which expresses the support from b for a, where α is a desired tolerance. "The more similar, the more close two values are, the more they support each other". The higher the tolerance is, the less we impose that the two values have to be closer to each other as absolute values.

$$Sup(a_i,a_j) = \begin{cases} 1 & \text{if} \quad |a_i - a_j| < \alpha \\ 0 & \text{otherwise} \end{cases} \tag{16.3}$$

If we were to aggregate a set of values and we wanted to order them in increasing order of support, "we compute for each value the sum of its support values with respect to all others values to be aggregated" [168]. Then, for each decision-maker opinion "we sum all the supports it has in order to obtain its overall support". These overall supports for a decision-maker's opinion are utilised as "the values of the order inducing variable". Below an example created by Yager to clarify the concept of a support vector is provided.

Let us assume that the threshold parameter is $\alpha = 0.4$ and that we have the following values to aggregate:

$$a_1 = 0.9,\ a_2 = 0.7,\ a_3 = 0.6,\ a_4 = 0.1,\ a_5 = 0$$

Using equation 16.3 above, we obtain the following support values:

$$\begin{array}{llll}
Sup(a_1,a_2) = 1 & Sup(a_1,a_3) = 1 & Sup(a_1,a_4) = 0 & Sup(a_1,a_5) = 0 \\
Sup(a_2,a_1) = 1 & Sup(a_2,a_3) = 1 & Sup(a_2,a_4) = 0 & Sup(a_2,a_5) = 0 \\
Sup(a_3,a_1) = 1 & Sup(a_3,a_2) = 1 & Sup(a_3,a_4) = 0 & Sup(a_3,a_5) = 0 \\
Sup(a_4,a_1) = 0 & Sup(a_4,a_2) = 0 & Sup(a_4,a_3) = 0 & Sup(a_4,a_5) = 1 \\
Sup(a_5,a_1) = 0 & Sup(a_5,a_2) = 0 & Sup(a_5,a_3) = 0 & Sup(a_5,a_4) = 1
\end{array}$$

The overall support for each a_i is computed by adding the support values for a_i. The support for each a_i is denoted as s_i:

$$s_1 = 2,\ s_2 = 2,\ s_3 = 2,\ s_4 = 1,\ s_5 = 1$$

Pasi and Yager [168] claim that it becomes evident that there are two clusters of similar values in s_i. Hence, the support function (equation 16.3) induces "a clustering of the arguments which can be controlled by the choice of the threshold parameter α in the aforementioned function $Sup(a_i,a_j)$". In the above example we can see that there are two clusters, 2's and 1's, with some ties of the support values. Yager claims that in order to address the ties we could impose a 'stricter' condition by setting $\alpha = 0.3$. Then, the new support vector would be:

$$s_1 = 1,\ s_2 = 2,\ s_3 = 1,\ s_4 = 1,\ s_5 = 1$$

This result enables us to "order the elements to be aggregated in the following increasing order of similarity" [168]:

$$\textit{Induced Similarity Order}, I = [0\ \ 0.1\ \ 0.6\ \ 0.9\ \ 0.7]$$

Pasi and Yager conclude that the use of an adequate support function *enables us to induce an ordering based on proximity*. This concept is key in understanding IOWA operators, as now it would be possible to generate a *majority-based aggregation* of the previous values a_i. As per Yager, "The selected IOWA operator should then correspond to the linguistic quantifier *most*. Let us recall the definition of the linguistic quantifier *most* presented

in equation (16.2)". This linguistic quantifier when used in equation (16.1) would derive the weighting vector $W = [0\ 0\ 0.4\ 0.4\ 0.2]$. Aggregating the vector I we obtain: $IW = 0.74$. However, the fact that the fifth element of the vector W is smaller than the fourth element, and despite the fact that this condition is coherent with "the interpretation of the weights as increase in satisfaction in having $i+1$ with respect to having i criteria satisfied", the expectation is that "in an aggregation with semantics of *majority* what would be expected is that the weights of the weighting vector are non-decreasing.". In fact, as in the induced order of the arguments the top value is the 'most supported' one from all the other values (the most representative) "it should be more emphasized than the others, or at least not less emphasized.". Pasi and Yager argue that a new strategy is required for the construction of the weighting vector that would contribute to generate a value more representative of a majority of the aggregated elements. The objective of this new strategy is to stress the most supported values in the resulting aggregation, i.e. the values shown on the right hand side of the vector of values participating in the aggregation do have more *influence* in the aggregation. As such, Pasi and Yager propose the following process for the construction of a weighting vector with non-decreasing weights.
Let us assume that the overall support (similarity) values computed for the n values to be aggregated $s_1, s_2, \ldots, s_n$. In order to calculate the non-decreasing weights of the weighting vector, the authors define the values $t_1, t_2, \ldots, t_n$ based on a modification of the $s_1, s_2, \ldots, s_n$ values:

$$t_i = s_i + 1.$$

In [168] the authors claim that by doing this manipulation, "the similarity of the value a_i with itself (similarity value equal to 1) is also included in the definition of the overall support for a_i. The t_i values are in increasing order, that is t_1 is the smallest value among the t_i. On the basis of the t_j values, the weights of the weighting vector are computed as follows":

$$w_i = \frac{Q(t_i/n)}{\sum_{i=1,\ldots,n} Q(t_i/n)} \tag{16.4}$$

"The value $Q(t_i/n)$ denotes the degree to which a given member of the considered set of values represents the *majority*". As such, equation 16.4 is the weights semantic we will apply to our aggregation problem.

Recall that we are looking for options to enhance our Hybrid Approach to the SA problem and that those improvements could come in the form of improvements in Accuracy, Precision and Recall. Others might come in a different presentation, looking more toward specific applications of our method either in isolation or in combinations with other techniques.

In this Chapter we address one of those specific applications, which is the scenario in which one might be interested in finding an aggregated value representing the opinion of the majority. In such a situation a proper aggregation mechanism must be found to integrate the opinion of several agents participating in a collective decision-making process.

Our Research Objective (Consensus in Sentiment Analysis driven by support-based majority): *Replace a number of n human agents in a collective decision-making process with the output of a number n of sentiment/opinion classification methods and aggregate these outputs with a method that semantically represents the concept of majority opinion ($n \geq 2$).*

Fig 16.2 depicts graphically the aggregation approach we propose to satisfy Our Research Objective. In the next section we will continue addressing the proposed IOWA aggregation operator that will address the consensus of opinion problem in sentiment analysis, centred around obtaining the intensity of the sentence's polarity.

16.5 The Proposed IOWA Approach to Sentiment Analysis (HACACO)

In this section we will describe how IOWA operators could be used to implement a fuzzy majority approach in the presence of recommendations (outputs) supplied by a number of classification systems. Full details about the science behind these equations and associated approach can be found in Section 13.1 and Section 13.1.1 of this document. We have called this method **H**ybrid **A**dvanced **C**lassification Method by **A**ggregation by **Co**nsensus (HACACO).

16.5.1 The Concept of Fuzzy Majority implemented using IOWA Operators

Constructing a *majority opinion* could be explained as "the collective evaluation of a majority of the agents involved in the decision problem" [168]. The following authors provide ample and detailed information about the OWA operators and its applications: Pérez et al. [170] and León et al. [127]; Yager [256]; Chiclana et al. [53] and Pasi & Yager [168]. Of particular interest are the three latter ones. In addition, Bordogna & Sterlacchini [31] and Boroushaki & Malczewski [33] provide very good examples of real applications of OWA operators.

16.5.2 Fuzzy majority in determining intensity of the polarity of predetermined subjectivity

Let us for a moment think of the problem of determining subjectivity polarity for a given sentence S_k using the recommendations of several systems. In a way, each method to be used and applied to the aforementioned sentence S_k, can be seen as a 'person' giving her opinion on whether the sentence S_k is positive or negative. In the end, we would like to collect all provided answers and come up with a sort of weighted mean of the inputs received. Basically, we would like to *aggregate the polarity* value of sentence S_k measured by using different classification methods. Hence, the final value will be the 'induced aggregation of the majority' of the subjectivity polarity of sentence S_k when one takes into consideration all the different contributions of all the participating methods. The different applied methods will issue their individual judgement on whether a sentence is positive or negative. We will call these methods $\{M_1, M_2 \dots M_n\}$. These techniques will arrive to their respective conclusions using their own appraisal strategies. Each method will have their own peculiarities. For instance, the Naïve Bayes method will classify a bag-of-words features, the fuzzy method will look at the level of belonging of a given particle to a given category (fuzzy set), the Maximum Entropy technique will apply its own philosophy, and so on. In fact, we would like to think that in addition to obtaining the aggregation already targeted, we could in the future incorporate the *level of trust* that we have on each method, ensuring that well-established proven or accurate methods carry more weight than the rest as we would do when considering the opinions of a number of people, depending on how much we *trust* each of them-. Figure 16.2 shows a graphical representation of the way the IOWA operator with the semantic 'most' is introduced, in order to achieve our pre-establish objective. The hybrid method

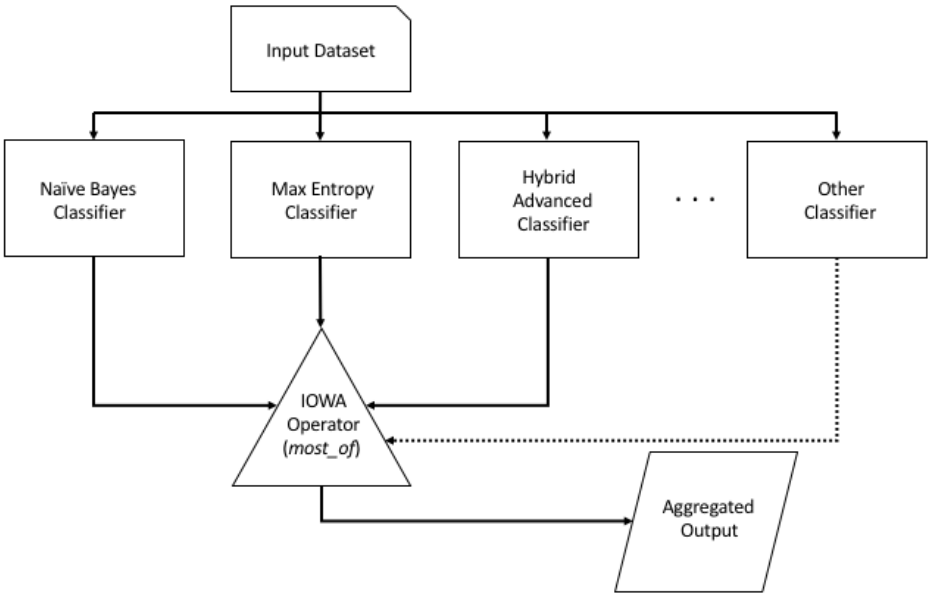


Fig. 16.2. *$IOWA_{most}$ Operator aggregating classifier methods outputs*

described in Chapter 14, can be improved with the incorporation of an IOWA operator with the semantic of the linguistic quantifier *most*, namely, $IOWA_{most}$, to handle the numerical output of three classification methods. In the example we are building, those methods are Naïve Bayes, Maximum Entropy and our Hybrid Method (HSC/HAC). Our $IOWA_{most}$ operator is capable of taking as input *any* number of outputs belonging to a variety of methods, with no theoretical limit to the number of methods' outputs that could be used. However, in order to make sense of the number n of methods to be aggregated the condition $n \geq 2$ is enforced.

16.5.3 Experiments results obtained applying IOWA$_{most}$ aggregation

In order to do a proper comparison, we will evaluate how the IOWA$_{most}$ operator performs when compared to both *Arithmetic mean* and *Median* [173]. However, we will firstly describe briefly the experimental methodology and the datasets utilised during the experimental phase.

16.5.4 Experimental Methodology - Summary

It is important to notice that the datasets for the experiments described in this section were not annotated at the beginning of the process, hence we had to introduce two additional tasks:

1. Assign intensity labels in $G = \{Poor; Slight; Moderate; Very; Most\}$ to the randomly selected 500 sentences.
2. Devise a criteria to discern the concept of what *consensus* would look like.

Both tasks above were executed by three individuals: the main researcher (the PhD candidate), the local thesis advisor (PhD) and a person knowledgeable in the English language (an English-major graduate). The latter did play the role of the expert (Linguist) when conflict resolution was required. For item number one above, the three individuals assigned a label to each of the 500 sentences according to their own criteria. The English-major resolved conflicts as they appeared, and a final decision on the assigned intensity polarity label was made. For item number two, the approach followed was to *attentively observe* how the combined classification scores were fused together, noticing whether the tested operators were capable to compensate for extreme values, close values and normally distributed occurrences. The expected effect of the IOWA Operator is precisely to *compensate* for the existence of outliers and to attempt to produce a number that reflects the semantic of the quantifier being used, in this case, the linguistic quantifier *most* that is driving the aggregation towards the opinion of the majority. As such, when the resulting aggregation was analysed, if the resulting number looked like one that had been obtained by *smart* aggregation/compensation, the score was considered to represent a successful case of consensus aggregation.

16.5.5 Datasets used

As mentioned in previous chapters, Pang and Lee [164] have published datasets that were utilised in SA experiments. As such, it seems adequate to use the Movie Review Dataset provided by Pang and Lee (available at: `http://www.cs.cornell.edu/people/pabo/movie-review-data/`). In order to use the output of all classifiers as an input to the IOWA$_{most}$ process all participating scores have been converted to the interval $[0,1] \in \mathbb{R}$, where S_k corresponds to any sentence in the test dataset and $m_i = \{m_1, m_2, \ldots, m_n\}$ represents the different classification methods i being aggregated ($n \geq 2$), then:

$$\text{IOWA}_{most}^{S_k}(m_1, m_2, \ldots, m_n) = \Theta^{S_k} \tag{16.5}$$

Once the aggregation with the semantic representing the *opinion of the majority* has been computed, then we must calculate to which intensity level that value Θ corresponds. For that, we use the classification method presented in Chapter 14, Sub-section 14.1.3, that estimates the intensity of the polarity of sentences, which is partially reproduced below. Recall that we used trapezoidal membership functions represented by the following 4-tuple (a, b, c, d):

$$\mu_{\tilde{A}}(x) = \begin{cases} 0 & \text{if} \quad x \leq a; \\ \frac{x-a}{b-a} & \text{if} \quad a \leq x \leq b; \\ 1 & \text{if} \quad b \leq x \leq c; \\ \frac{d-x}{d-c} & \text{if} \quad c \leq x \leq d; \\ 0 & \text{if} \quad d \leq x. \end{cases} \tag{16.6}$$

Specifically, the following granules on the perception of the *intensity of the polarity*, either *positivity* or *negativity* of a given sentence S are suggested: $G = \{Poor; Slight; Moderate; Very; Most\}$, with the following 4-tuples (MF means *membership function*):

- MF (Poor): (0, 0, 0.050, 0.150)

- MF (Slight): (0.050, 0.150, 0.250, 0.350)
- MF (Moderate): (0.250, 0.350, 0.650, 0.750)
- MF (Very): (0.650, 0.750, 0.850, 0.950)
- MF (Most): (0.850, 0.950, 1, 1)

The aggregated value Θ previously computed in equation 16.5 will take on the value x in equation 16.6 and in consequence a proper linguistic label belonging in G will be generated. This value represents the polarity intensity (how positive or how negative) of a given sentence S_k ($\mu_{\tilde{A}}(\Theta^{S_k}) \in G$).

16.5.6 Comparison criteria

The comparison we are attempting to make might be challenging to achieve as the intensity of the polarity of a sentence would be vague in nature, as opposed to crisp. We are trying to see which method is semantically closer to the opinion of the majority among the participating methods. The classification of a sentences as belonging to one of the granules we defined in section 14.1.3.1, $G = \{Poor; Slight; Moderate; Very; Most\}$, is rather a subjective exercise. The datasets used count each (positive occurrences and negative occurrences) with 5,331 sentences. We have annotated 500 sentences, approximately 10%, assigning each of them a value $v_k \in G$, that has been estimated by looking at the classification outcomes of the three classifiers we are utilising as inputs and estimating a linguistic label in G that is representative of the opinion of the majority.

16.5.7 Non-OWA Aggregation - The outputs of the three classification methods combined without the application of the IOWA operator

Before we applied the IOWA_{most} operator, we tested the idea of combining directly the results of the three chosen methods. The outcomes, which are summarised below, are not as good as those obtained by using the IOWA operator, as it will become evident later on. This fact, basically, shows that the IOWA operator does a much better job at aggregating the individual outcomes of the three aforementioned techniques, by giving more weight to the leaning opinion of the majority. In essence, by weighing properly the advises of the three methods -NB, ME and our Hybrid Advanced Approach- we do obtain a more realistic aggregation effect that represents what the majority stands for.

The first method we used -that does not utilise the IOWA operator- is *Median*. Below in Table 16.3 shows the associated Performance Indexes for *Median*. The second method we used -that does not utilise the IOWA operator- is *Arithmetic*

Represents opinion of the majority	337
Does **not** represent opinion of the majority	163
% of success	67.40

Table 16.3. Method I: *Median*

mean. With this technique we simply obtain an arithmetic average of the outputs of the involved systems. Table 16.4 shows the associated Performance Indexes for *Arithmetic Mean*.

Represents opinion of the majority	388
Does **not** represent opinion of the majority	112
% of success	77.60

Table 16.4. Method II: *Arithmetic Mean*

16.5.8 IOWA Aggregation - Combination of outputs of the three classification methods using operator IOWA$_{most}$

These results that we have obtained, simply re-enforce the applicability of doing aggregation using the IOWA operator with a semantic associated to a specific linguistic quantifier (in this case, *most*). The results of using IOWA$_{most}$ are shown in Table 16.5 and Table 16.6, whilst Table 16.7 presents a comparison of results.

Represents opinion of the majority	500
Does **not** represent opinion of the majority	0
% of success	100.00

Table 16.5. IOWA$_{\text{most}}$ operator - Tolerance = 0.30

Represents opinion of the majority	500
Does **not** represent opinion of the majority	0
% of success	100.00

Table 16.6. IOWA$_{\text{most}}$ operator - Tolerance = 0.50

Notice that the aggregation results obtained using the IOWA operator are much more compelling than the rest. Especially,

Classification Method	**Median**	**Arithmetic Mean**	**IOWA**$_{most}^{tolerance=0.3}$	**IOWA**$_{most}^{tolerance=0.5}$
% of success	67.40%	77.60%	100.00%	100.00%

Table 16.7. The three aggregating methods - *Performance Indexes Compared*

because IOWA$_{\text{most}}$ *always* represents the targeted majority, as a consequence is the best option when compared to the other two methods tested. The main difference between the results obtained when using different tolerance values (0.3 and 0.5) when IOWA$_{most}$ is applied, is not in whether the outcome will distance itself from representing the opinion of the majority, but rather in which *linguistic label* in G a specific sentence will be assigned to. Depending on the majority value calculated a sentence classified as 'Moderate' with a tolerance of 0.3 could now be labelled as 'Very' in terms of intensity, when the tolerance value changes to 0.5. In reality, the lower the *tolerance*, the more demanding the IOWA operator is on how closely the values in the aggregation support each other.

16.5.9 Examples of applying the IOWA$_{\text{most}}$ operator to specific members of the dataset

In this section we will present examples of the application of the IOWA$_{\text{most}}$ operator to several sentences. The examples include the output of three different classification methods, m_1, m_2, m_3, where their outputs belong in the interval $[0, 1]$.

Example 16. $(m_1\, m_2\, m_3)$ = *(0.959112 0.500030 1.00000).*
Arithmetic mean = 0.819716
Median = 0.959112
IOWA (Tolerance = 0.50) = 0.819717
With a tolerance of 0.5, which does not enforce strict support among the values to be aggregated, all the elements contribute to the aggregation, generating a value that is extremely close to the arithmetic mean.

Example 17. $(m_1\, m_2\, m_3)$ = *(0.564631 0.508914 1.000000).*
Arithmetic mean = 0.691181
Median = 0.564631
IOWA (Tolerance = 0.50) = 0.536773
With a tolerance of 0.3, which does enforce a stricter support among the values to be aggregated (in this case the first two

values), the value that represents the sentiment of the majority is closer to the elements with higher support: 0.564631 and 0.508914; as a consequence the generated value is not that close to the arithmetic mean, neither the mean, although it is closer to the latter.

Example 18. $(m_1\, m_2\, m_3)$ = *(0.989550 0.682592 0.600000).*
Arithmetic mean = 0.757380
Median = 0.682592
IOWA (Tolerance = 0.30) = 0.641296
With a tolerance of 0.3, again the IOWA aggregation with the semantic most, generates an aggregation between 0.682592 and 0.600000, which are supporting each other, representing again the opinion of the majority.

16.5.10 The role of the threshold parameter in the calculation of the support vector in IOWA

We have mentioned before that the IOWA operator used to generate the aggregation is formulated in such a way that a tolerance input is provided during the aggregation process. Let us look at the results obtained when we map the tolerance parameter against the polarity intensity classification of the test dataset. The *tolerance* value (α) is used in the calculation

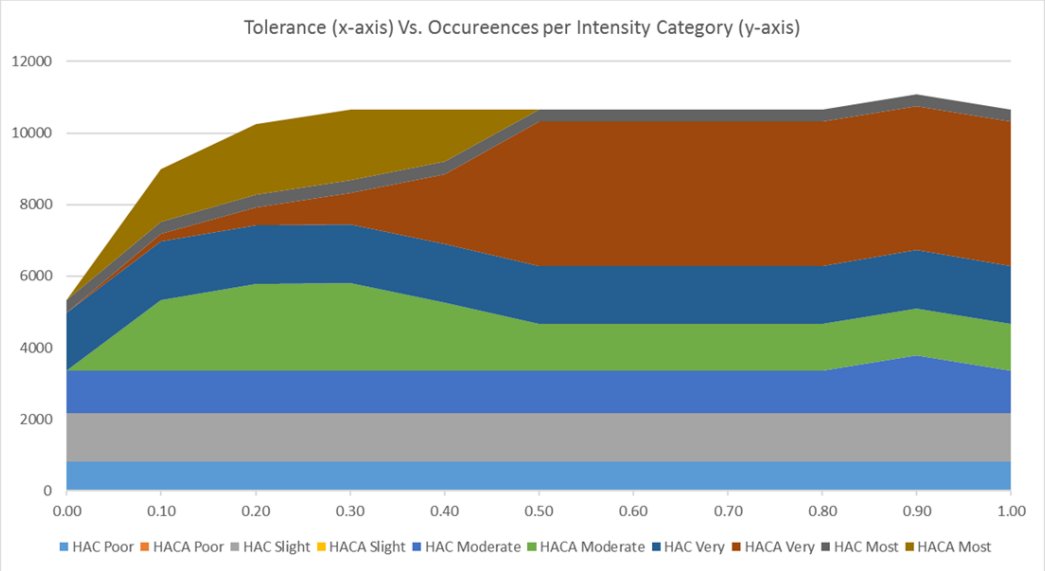


Fig. 16.3. Tolerance vs. Polarity Values

of the support vector mentioned above in equation (16.3). The farther apart a_i and a_j, the higher the value of $|a_i - a_j|$. The higher the tolerance value α becomes, the more likely the support function $Sup_\alpha(a_i, a_j)$ will take a value of 1, which means that both values a_i and a_j will be part of the aggregation being performed. This translates into a situation in which: if a value a_i is too far apart from the rest of values a_j $(j \neq i)$ so that its corresponding distances to the them exceed the defined tolerance for the support vector, $|a_i - a_j| > \alpha$, then that value contribution to the aggregation process will be very low as it will have assigned a value of $t_i = 1$. A tolerance value $\alpha = 1$ means that all values considered in the aggregation will contribute equally to the collective aggregated value regardless of how close they are to each other, and the IOWA^1_{most} will coincide with the *arithmetic mean*. On the other hand, a tolerance value of, let us say, 0.1 will imply that the considered values for aggregation will have to be very close to each other to have a high support and higher contribution to the collective aggregated value. In our experiment, Fig. 16.3 shows that the highest the value of the *tolerance* parameter is, the more the intensity polarities are distributed towards the linguistic labels 'very' and 'most', which are located towards the very right-end of the unit interval $[0, 1]$. Notice as well that the tipping point in the x-axis is the value 0.5 (in Fig. 16.3, *HAC* stands for Hybrid Advanced Classification, whilst *HACA* means Hybrid Advanced Classification with IOWA Aggregation, that are methods we introduced in Chapter 13 and developed in Chapters 15 and 16).

16.6 Chapter Summary

The method we have introduced in this Chapter enables us to produce the opinion of the majority by using an IOWA operator reflecting the semantics of the fuzzy quantifier *most*. The experimental results show that the proposed *aggregation by consensus* method is a much better option than two other approaches (Arithmetic Mean and Median) when the objective is to obtain an opinion that represents the majority.

In this chapter we have presented an enhancement to our proposed hybrid method that is based on *aggregation by Consensus*. In the next chapter (Chapter 17) we will present some other research options that were explored but that did not come to fruition.

Chapter 17

Other Paths Explored

"It's hard to build models of inflation that don't lead to a multiverse. It's not impossible, so I think there's still certainly research that needs to be done. But most models of inflation do lead to a multiverse, and evidence for inflation will be pushing us in the direction of taking [the idea of a] multiverse seriously."

Alan Guth, Theoretical Physicist and Cosmologist. Developed the idea of cosmic inflation in 1979.

17.1 Other research paths explored but not pursued

In our research effort, there were a number of alternatives that were explored and provided invaluable learnings, even if at the end, they did not become part of the proposed research solution. A number of options were explored -some of them in detail- and then abandoned either because the outcomes produced were not successful or because we did not think convenient at the time to keep exploring those paths.

17.1.1 Polarity and Polarity Intensity Classification in one step

Our proposed solution HSC/HAC works in two steps, with Step HSC calculating the subjectivity polarity and Step HAC supplying the polarity intensity. We considered doing it all in one step with an aggregative cross-uninorm as presented in section 15.2, equation 15.4. Having to perform polarity and polarity intensity computations in one step may be cumbersome if we attempt to use methods that require as input *only* numbers. That was the case of our approach, as the participating elements in the calculation were the polarity scores of the words making up a sentence. The main two challenges were:

- The attributes Positive (PSC) and Negative Polarity (NSC) in the lexicon had to be translated from the unit interval $[0, 1]$ to the interval $[-1, 1]$.
- We were not able to incorporate critical elements that are part of the HSC/HAC approach, like the fact that there is an apparent hierarchy of importance among the opinion-conveying part of speech elements already identified (adjectives, adverbs, verbs and nouns). We believe that when it comes to semantic orientation calculation, the strata approach mentioned in Chapter 14, Table 14.4 is effective. A possible future effort would be to use a one-step process as described above, but combining the use of the cross-uninorm with an algorithm that acknowledges as well the level of importance among members of the selected Part-of-Speech class.

These topics above mentioned were covered as well in Appel et al. [16].

17.1.2 Incorporation of the Concept of Emotion

In Chapter 11, we discussed the concept of emotions from a psychological point of view. Specifically, in Section 11.1 we discussed the Ortony - Clore - Collins (OCC) model. Initially, we considered using the OCC model in the formulation of our proposed solution, but after looking at the details and doing some crude modelling inspired in the structure introduced by OCC, we did not find any fundamental difference in the results obtained when calculating subjectivity polarity. Using the OCC Revisited model described in section 11.2 (Fig. 11.2), we went through the effort of naming and defining types, some properties and interrelationships of the entities in our lexicon. However, the impact produced was not significant. Having said that, we did not go too deep into this area, and perhaps there is room for more research to be conducted in this field, possibly as further work.

17.1.3 VSM & PMI as Lexicon Quality Enhancers

In Section 10.2 we discussed the possibility of using the VSM and the PMI approaches as options to introduce valuable data into the Sentiment Lexicon. How could we increase the knowledge of our Sentiment Lexicon (SL) / Opinion Lexicon (OL)? As shown in Table 17.1, we had a number of opinion-conveying words for which we did not find a *synset* in SentiWordNet. Hence, we thought about the possibility of deriving some additional data that could be included in our Sentiment Lexicon. We proposed back then a lexicon entry as described below:

#(**Word SOL PoS PositiveScore NegativeScore CalculatedObjectivityScore SemanticOrientation MaxDist MinDist UpdateCounter**)

The labels devised for the particles that were supposed to be part of our sentiment lexicon were:

1. Word: word in the lexicon (entries)
2. SOL: Semantic Orientation label (either POS or NEG)
3. PoS: Part of Speech (n=noun; v=verb; a=adjective; r=adverb; s=adjective satellite)
4. Positive Score (PSC), as taken from SentiWordNet [79]
5. Negative Score (NSC), as taken from SentiWordNet [79]
6. Calculated Objectivity Score (COBJ)
7. Calculated Semantic Orientation (CSOR)
8. Calculated Max Distance to Positive Seeds (MAXDIST)
9. Calculated Min Distance to Negative Seeds (MINDIST)
10. Update Counter (UPDC): to keep track of every time a given entry in the lexicon is updated)

Items 7, 8 and 9, 'CSOR', 'MAXDIST' and 'MINDIST', were obtained using the methodology discussed in section 10.2, making full use of the VSM/PMI approaches.

As such, in terms of completion (how well populated it is) the state of the opinion lexicon is shown in Table 17.1. Nevertheless, notice the low percentage of completion for CSORs, MAXDISTs and MINDISTs, which is between 5.0% and 6.5%. What this means is that the *Word x Word* matrix described in section 10.2, which was used as part of our Vector Space Model of Semantics strategy, did not contain many of the words that were already part of our opinion lexicon. Having said that, the PoS, PSC, NSC and COBJ levels of completion exceed the 60% mark, which is a rather acceptable level. As a consequence, we made the decision of continuing using those values considered as *key* to our lexicon, which were *PSC, NSC* and *PoS*.

We never utilised any of the two techniques (VSM & PMI) in the final proposed solution, as described in Chapter 14. In closing, the reasons why we opted out of creating these attributes (CSOR, MAXDIST & MINDIST) as part of the lexicon, were:

	Positive Lexicon	% completeness	Negative Lexicon	% of completeness
No. of Entries	2,006	n/a	4,783	n/a
Entries w/ PoS	1,320	65.80%	2,878	60.17%
Entries w/ PSC	1,320	65.80%	2,878	60.17%
Entries w/ NSC	1,320	65.80%	2,878	60.17%
Entries w/ COBJ	1,320	65.80%	2,878	60.17%
Entries w/ CSOR	147	7.33%	243	5.08%
Entries w/ MAXDIST	249	12.41%	307	6.42%
Entries w/ MINDIST	249	12.41%	307	6.42%
Entries w/ UPDC	n/a	n/a	n/a	n/a

Table 17.1. Opinion/Sentiment Lexicon Status - Completeness of data (attributes)

- There were many terms (words) for which we could not calculate these attributes. This situation happened as many terms were not available in the public corpus that was used to generate the PMI matrix.
- After experimentation, the impact they made and their contribution to the final results were not as influential as initially thought. We opted instead for another alternative, like using a Word Dictionary of Frequencies of Occurrences (Section 10.4).

Fundamentally, the idea behind using a computed maximum and a minimum distance from a given word *X* to one of the two sets of paradigms for positive and negative meaning words, as described by Turney [212] and Turney & Littman [213, 214], was to be able to know *how far or near* a given word *X* was from holding a specific subjectivity polarity. The initial intention was being capable of finding the polarity of a word when the latter was not present in the sentiment lexicon. As in many cases the needed word was not in the calculated PMI Matrix, we ended up opting out of this strategy.

17.2 Chapter Summary

In this chapter we have discussed a number of research options that we initiated exploring but did not pursue on implementing. In the next chapter, which is included in Part V, we will share our conclusions and will discuss some possible further work options.

Part V

CONCLUSIONS & FURTHER WORK

Chapter 18

Conclusions & Further Work

The Good, the Bad and the Ugly. (Il buono, il brutto, il cattivo.)

Title of the 1966 Italian epic Spaghetti Western film directed by Sergio Leone.

Some portions of the contents of this chapter were utilised in articles published by the author. See references [10, 11, 12, 13, 14, 15, 16, 17, 18]. The following three sections will cover individually the conclusions and further work aspects of the three approaches firstly described in the *Abstract* of this document and elaborated in detail in Chapters 14 through 16:

1. The design of a Hybrid Classification Model based on: (i) the positive contribution that NLP tools, semantic rules and a solid opinion lexicon can have in identifying sentiment polarity; and (ii) the concept of *graduality* expressed through fuzzy sets. Through extensive experimental work, this method has been proved capable of extracting sentiment from text whilst being a much better performer than established Supervised Machine Learning techniques -namely, Naïve Bayes and Maximum Entropy (ME)- when the latter are utilised respectively as the only classification method being applied.

2. The introduction of cross-ratio uninorms to effectively fuse the classification outputs of several algorithms producing a compensatory effect, as an alternative approach to the existing two approaches to deal with those cases when a lexicon-based SA method cannot produce a classification output (there are terms required for the analysis that are absent from the sentiment lexicon): use the services of a Word-frequency Dictionary or add to the lexicon (off-line) the missing words required to complete the analysis.

3. The utilisation of a specific Induced Ordered Weighted Averaging (IOWA) operator as a tool to model the opinion of the majority (consensus) when the outputs of a number of classification methods are combined together.

Overall, the main formulated hypothesis and associated research questions discussed in Chapter 5 (Section 5.2) have been proven and answered. Our experimental results have shown that:

1. In fact, a *hybrid approach* as described in our Hypothesis 1, is indeed well equipped to model subjectivity polarity determination and polarity graduality determination in Sentiment Analysis / Opinion Mining at the sentence level.

2. *Lexicon-based methods* are capable of delivering similar *precision* to the one provided by more traditional *Supervised Machine Learning* techniques in the determination of polarity subjectivity in Sentiment Analysis.

3. *Fuzzy methods* are an excellent tool to model polarity intensity in Sentiment Analysis by introducing gradualness (graduality) via fuzzy sets.

4. *Semantic rules* are a very good mechanism for computing semantic orientation in both, words and sentences.

The following sections will provide conclusions that are specific to the three proposed methods introduced in this PhD Thesis in Chapters 14, 15 and 16.

18.1 Hybrid Classification (HSC/HAC)

As mentioned in Chapter 14, Section 14.5, our proposed hybrid system (HSC/HAC) works very well at the sentence level showing high scores for the selected performance indicators. Our expectation is that the quality of the content of SentiWordNet, or more recent tools like SenticNet [45], should continue to improve with time. Those enhancements will contribute to the betterment of our proposed hybrid solution as it will reflect positively in the quality of our opinion lexicon. In theory, as time passes, both SentiWordNet and our proposed opinion lexicon should become better and more complete. The ability to incorporate new terms into the current opinion lexicon in an expedite way is another benefit provided by our proposed solution.

In essence, hybrid techniques can play an important role in the advancement of the Sentiment Analysis discipline by combining together a number of elements that tend to produce better results. Similar results, in terms of combining techniques effectively, have been reported by other researches [176].

By carefully analysing the sentences processed using the aforementioned classification methods, we are capable of extracting the main characteristics in those sentences that posed a problem for our classification method. If we group together the cases that our system considered challenging to classify properly, we find the following traits:

- The absence in our Sentiment Lexicon of words (terms) that convey opinion is a challenge for our system. If for a given sentence, at least one term is available in the lexicon, our system will attempt to generate a semantic orientation based on that existing term, but if none is available the sentence is classified as a NSO (no Semantic Orientation available).
- The use of jargon, argot, idiom and/or lingo is hard to deal with, and sometimes it misguides the system in classifying opinions properly.
- Imagery, metaphors, similes, sarcasm, humour and other language figures that rely on previous knowledge and/or context represent a challenge for our system. For future research, a starting point would be the work of Justo et al. [109].
- Double negation offers difficulties that we must continue to study and improve.
- In the presence of very complex paragraphs the precision of our proposed hybrid method is negatively impacted.

In terms of future research work, we believe there are a number of avenues that should be pursued in the short-term:

- Create an automatic real-time interface, via API, with SentiWordNet or functionally equivalent tool (see next item) to search dynamically for polarity and PoS tagging updates for all terms of interest.
- Investigate the possibility of using SenticNet [45] as a source of a more mature and comprehensive set of semantic attributes to enrich our own lexicon -or to replace it. The concept-level sentiment analysis approach introduced by Poria et al. [176], sentic patterns, and its dependency-based rules could provide a broader semantic coverage than the one we currently enjoy with SentiWordNet.
- Work on an algorithm to incorporate context in the tagging/parsing process and in the sentiment-determination modules in order to improve the ability of the system to deal with humour, metaphors, similes, sarcasm and irony (an initial approach could be the utilisation of context-sensitive grammars during the parsing process).
- Port the proof-of-concept prototype code from Scheme to C, C++, C# or Python, in order to increase efficiency and integration capabilities.
- Continue developing a *computing with sentiments* approach, using as a foundation the method presented in Chapter 14.
- Introduce the use of Induced OWA operators [258] as an aggregation mechanism in the calculation of the semantic orientation of sentences. This type of operator could be used to ensure that specific values among those elements considered would drive the aggregation according to a predefined operator. This aggregation could be performed at both levels, the words participating in a sentence and the sub-sentences making up a full sentence or paragraph. Work on this approach has already started and we have as an initial outcome the work presented in Chapter 16.

18.1.1 Examples of sentences

There are some sentences that has proven to be challenging for our hybrid system to elucidate properly. We will show some examples that represent classes of sentences that have something in common that make them challenging for our system to evaluate properly.

Examples of sentences Labelled as *Negative* that were wrongly classified

Sentence 1 (from Twitter): "Played with an android google phone. The slide out screen scares me. I would break that fucker so fast. Still prefer my iphone."
The author of this sentence indirectly point to a negative aspect of the android google phone by saying that she could break its slide out screen easily; hence, she prefers her iphone phone. Our system does not comprehend the subtlety.

Sentence 2 (from Twitter): "Was just told that Nike layoffs started today."
This sentence is part of a thread related to unemployment in the USA. Our system does not see this sentence as Negative, as it seems to be implying simply that lay-offs had started for an object called 'Nike'.

Sentence 3 (from MoviesReviews): "While the performances are often engaging this loose collection of largely improvised numbers would probably have worked better as a one hour TV documentary."
The suggestion expressed by the author of the sentence that the movie would have worked better as a TV documentary does not seem to carry enough weight to sway the classifier from identifying this sentence as *positive*. Let us look at the structure of a sentence using the particle 'while': While $< Sentence\ X > < Sentence\ Y >$. According to one of the Semantic Rules (Rule 14) 14.1.2 we defined and applied in our system, in the presence of sentences using the particle 'while', the semantic of the whole sentence is defined by the semantics of $< Sentence\ Y >$. The sub-sentence 'would probably have worked better as a one hour TV documentary' is not conveying a real negative connotation but it rather suggests a preference.

Examples of sentences Labelled as *Positive* that were wrongly classified

Sentence 4 (from Twitter): "Lebron is the boss."
Lebron James is an American professional basketball player. Our hybrid system is unable to classify this sentence as positive as there is not context for it to know who Lebron James is. As such this sentence is classified to be an objective expression stating the Lebron is the boss.

Sentence 5 (from Twitter): "Let is go Cavalliers."
Firstly, the intended expression is "Let us go Cavalliers" instead us "Let is go Cavalliers". However, even entering the right sentence our system fails to recognise that the Cavalliers is a basketball team and that the author of the sentence is encouraging them to win an upcoming game. Again, the system sees it an an objective sentence.

Sentence 6 (from MoviesReviews): "Steers turns in snappy screenplay that curls at the edges and it is so clever you want hate it, but he somehow pulls it off."
One of the semantic rules in our hybrid system calls for taking the semantic after a 'but' particle as the semantic orientation of a sentence, ignoring the proceeding sentence. In this case the sub-sentence that follows the 'but' particle, "he somehow pulls it off", uses an expression to point to the fact that the author of the movie's script manages to deliver -in the end- something very good. Our system did not understand the meaning of 'pulls it off' as something positive. In another sentence using the 'but' particle, though, our system succeeds: "The film started with a cloudy sky but suddenly the sun came out in all its splendour making it for a lovely day". In this latter case, the sub-sentence after the 'but' particle is clearer to our system, (...the sun made it for a lovely day), which provides a solid positive meaning for the sub-sentence. In the former case, the sub-sentence after the 'but' used a *lingo expression* whilst the latter one employed a well constructed sentence that was parsed and tagged properly.

Examples of sentences correctly classified

Sentence 7 (Positive): "The Rock is destined to be the 21st century new Conan and that he is going to make a splash even greater than Arnold Schwarzenegger, Jean Claud van Damme or Steven Segal."
Sentence 8 (Positive): "Emerges as something rare an issue movie that is so honest and keenly observed that it does not feel like one."
Sentence 9 (Negative): "It is so laddish and juvenile only teenage boys could possibly find it funny."
Sentence 10 (Negative): "A visually flashy but narratively opaque and emotionally vapid exercise in style and mystification."

In general, our proposed hybrid system works very well with a high level of accuracy and precision, with the exception of some cases where the sentences at hand belong in one of the categories described in the list just discussed above.

18.2 HACACU: HSC/HAC plus Cross-ratio Aggregation

Cross-ratio uninorm operators can certainly play a significant role in aggregating the opinions of a number of classification systems in a more balanced way, compensating when required for specific data traits, as discussed in Section 15.2, behaving like a *conjunctive*, *disjunctive* or *compensatory* operator as required.

If we recall our initial motivation, for those cases when a lexicon-based SA method cannot produce a classification output (there are terms required for the analysis that are absent from the sentiment lexicon) we could as an alternative option use the services of a Word-frequency Dictionary or add to the lexicon (off-line) the missing words required to complete the analysis. However, adding words off-line is expensive because it requires the knowledge of an expert and it is time consuming, which means that the method cannot produce an answer immediately, i.e. it prevents its automation. The other alternative method involves the creation of the Word-frequency Dictionary, which is computationally expensive, $O(n^2)$, for creation and traversing. In contrast to these approaches, the alternative cross-ratio uninorm approach is easy to implement and is computationally inexpensive. In addition, it performs much better than the Word-frequency Dictionary. Although the proposed cross-ration uninorm approach performs slightly worse than the off-line addition of the missing words to the lexicon, it has the advantage of being less time consuming and allows to automate the whole SA process. This situation provides us with options:

1. If one can afford the costs of adding missing terms to the sentiment lexicon and it is possible to wait for a more precise answer, then HSC as presented in Chapter 14 is the best choice.
2. If one is urged to provide an answer immediately, then there is the convenient alternative of using the cross-ratio uninorm approach presented in this Chapter.

It is a matter of a compromise, between off-line extra time to look for the required words and getting them in the lexicon, and providing an immediate answer with a potential slight lesser accuracy. In order to put things in perspective, let us remember that many commercial software packages in the realm of machine learning provide the option of utilising the so-called ensemble averaging methods. Typically, this technique works by combining previously created methods in order to produce a desired output. Usually the steps are: (i) obtain the outputs of N methods, (ii) separately, train each model, and (iii) combine the method outputs and average their values. In some cases, a slightly more complex approach is followed, and the ensemble averaging is performed as $\tilde{y}(x;\alpha) = \sum_{j=1}^{N} \alpha_j y_j(x)$, where each method output is y_j, α values represent a set of weights, and N is the number of methods being considered. This version corresponds to a weighted sum instead of a mere average. However, as it has been shown in the experimental results section, the proposed uninorm based method performs better than standard average functions. As such, a uninorm approach would provide better results. In addition, if it is true that the proposed technique was introduced as a complement to the lexicon-based classification method presented in Chapter 14, we believe that the cross-ratio uninorm described here could be utilised as well as a more efficient and focused ensemble method where the semantic of the aggregation represents a symmetric aggregative effect.

In terms of further work, there are two avenues that could be pursued in the short-term:

- In addition to obtaining the aggregation already mentioned among the classification methods, we could incorporate the level of trust that one has on each method $\{M_1, M_2, \dots M_n\}$, ensuring that those better established and proven methods carry more weight (as we would do when pondering the opinions of a number of people, depending on how much we *trust* each of them).
- There are multiple uninorm operators available (which stem from the research field of *decision making theory*) and some of them are highly flexible depending on the selection of the appropriate quantifier or function. As such, we would like to explore further additional options that could potentially provide even better results. Especially, around the possibility of utilising equation 15.4 in aggregating words polarity scores and sub-sentences polarity scores in an approach to produce a semantic orientation score without the need for using polarity labels (Positive/Negative) and using the polarity interval $[-1, 1]$ instead of $[0, 1]$. Of course, such changes would require the introduction of modifications to the existing sentiment lexicons.

18.3 HACACO: HSC/HAC plus Consensus Aggregation

Induced Ordered Weighted Averaging (IOWA) operators can certainly play a significant role in aggregating the opinions of a number of sentiment classification systems. The aforementioned operator works by producing a value that get significantly closer to the collective opinion of the participants. The IOWA_{most} operator we have presented in this article conveys the semantic of the opinion of the majority, which is represented by the linguistic quantifier *most*. Its performance in identifying the *intensity of the opinion of the majority*, according to our experiments, surpassed the one exhibited by *Arithmetic Mean* and *Median* techniques.
The results we have obtained are sensible as our IOWA_{most} operator produce results that gravitate towards the opinion of most of the input values being processed. In essence, IOWA_{most} produces a larger pull towards the values that support each other, driving the results in the direction of what the majority stands for.

In terms of further work, we believe there are some avenues that could be pursued in the short-term:

- Investigate other OWA operators that could potentially produce a better aggregation representing the semantic *majority opinion*.
- In addition to obtaining the aggregation already mentioned among the classification methods, we could incorporate the level of trust that one would have on each method $\{M_1, M_2, \dots M_n\}$, ensuring that those better established, respected and proven methods carry more weight in the aggregation, as one would do when pondering the opinions of a number of experts, depending on how much one *trusts* each of them.
- Utilise the OWA measure *Dispersion*, $Disp(W)$, which calculates the degree to which all aggregates are used equally in the resulting final aggregation [169]. The idea would be to gain a deeper understanding on how the support vector is configured to contribute to the semantic of a majority opinion, depending on the data values participating in the aggregation.

18.4 Evolution of the proposed solutions

It would be rather interesting to see how our proposed hybrid system evolved in time as new features were incorporated. In Table 18.1, we present the full progression of the methods we have introduced in this doctoral thesis document.

18.5 Chapter Summary

In this chapter we have shared our conclusions, which brings us to the end of this thesis report. The next components of this report, collected as *appendices*, are summarised below:

- Appendix A: Scientific contributions of the author during his PhD studies
- Appendix B: Prototype Outputs

Method	**Acronym**	**Description**	Precision (Movie DB)	Precision (Twitter DB)
Hybrid **S**tandard **C**lassification	HSC	Produces a Polarity Classification using our hybrid method	0.7278	0.8424
Hybrid **A**dvanced **C**lassification	HAC	Generates a polarity intensity classification	Precision = 0.7278 (same as HSC). [Intensity Correctness: Poor = 0.81, Slight = 0.89, Moderate = 0.93, Very = 0.91, and Most = 0.87]	Precision = 0.8424 (same as HSC) [Intensity Correctness: N/A]
Hybrid **A**dvanced **C**lassification with **A**ggregation by **C**ross-ratio **U**ninorm	HACACU	Incorporates Cross-ratio Uninorm Aggregation to compensate for situations where it is not possible to output a lexicon-based classification	0.7673	0.8479
Hybrid **A**dvanced **C**lassification with **A**ggregation by **Co**nsensus	HACACO	Adds the ability to find consensus among a number of classification methods (achieve 100% accuracy modelling consensus)	N/A	N/A

Table 18.1. Evolution of the Proposed Method

- Appendix C: Scheme Code - SA Hybrid System (Proof of Concept)
- Appendix D: Data Preparation & Pre-processing
- Appendix E: Classification Output for Naïve Bayes & Maximum Entropy
- Appendix F: Samples of outputs of Syntactic Conversions Programs
- Appendix G: Examples of the application of Semantic Rules & Negation

References

[1] Steven Abney. Statistical methods and linguistics. In Judith Klavans and Philip Resnik, editors, *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*. The MIT Press, Cambridge, MA, 1996.

[2] Steven Abney. *Semisupervised Learning for Computational Linguistics*. Chapman & Hall/CRC, 1st edition, 2008. ISBN 978-1-58488-559-7 (Hardcover).

[3] Charu C. Aggarwal and Chengxiang Zhai, editors. *Mining Text Data*. Springer, 2012. ISBN 978-1-4419-8462-3.

[4] Muntaha Ahmad and Ajay Rana. Fuzzy sets in Data mining – A Review. *International Journal of Computer Technology and Applications (IJCTA)*, 4(2):273–278, Mar-Apr 2013. ISSN 2229–6093.

[5] A.V. Aho and J.D. Ullman. *The Theory of Parsing, Translation and Compiling, Vol I: Parsing*. Englewood Cliffs, N.J.: Prentice Hall, June 1972.

[6] A.V. Aho and J.D. Ullman. *The Theory of Parsing, Translation and Compiling, Vol II: Compiling*. Englewood Cliffs, N.J.: Prentice Hall, 1973.

[7] Hejab M. Alfawareh and Shaidah Jusoh. Resolving Ambiguous Entity through Context Knowledge and Fuzzy Approach. *International Journal on Computer Science and Engineering (IJCSE)*, 3(1):410–422, January 2011. ISSN 0975-3397.

[8] Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2nd edition, 2010. ISBN 026201243X, 9780262012430.

[9] Kalaiarasi Sonai Muthu Anbananthen and Ayoub Mohamed H. Elyasir. Evolution of opinion mining. *Australian Journal of Basic and Applied Sciences*, 7(6):359–370, 2013. ISSN 1991-8178.

[10] Orestes Appel, Francisco Chiclana, and Jenny Carter. Main Concepts, State of the Art and Future research Questions in Sentiment Analysis. *Acta Polytechnica Hungarica - Journal of Applied Sciences*, 12(3):87–108, May/June 2015. ISSN 1785-8860. doi: 10.12700/APH.12.3.2015.3.6. URL `http://dx.doi.org/10.12700/APH.12.3.2015.3.6`.

[11] Orestes Appel, Francisco Chiclana, Jenny Carter, and Hamido Fujita. *A Hybrid Approach to Sentiment Analysis with Benchmarking Results*, pages 242–254. Springer International Publishing, Hamido Fujita, Moonis Ali, Ali Selamat, Jun Sasaki and Masaki Kurematsu: Editors, Cham, 2016. ISBN 978-3-319-42007-3. doi: 10.1007/978-3-319-42007-3_21. URL `http://dx.doi.org/10.1007/978-3-319-42007-3_21`.

[12] Orestes Appel, Francisco Chiclana, Jenny Carter, and Hamido Fujita. *A Hybrid Approach to Sentiment Analysis*, pages 4950–4957. IEEE: Proceedings of 2016 IEEE Congress on Evolutionary Computation (CEC): IEEE World Congress on Computational Intelligence (IEEE WCCI-2016), Vancouver, Canada, 24-29 July 2016, 2016. ISBN 978-1-5090-0623-6/16.

[13] Orestes Appel, Francisco Chiclana, Jenny Carter, and Hamido Fujita. A hybrid approach to the sentiment analysis problem at the sentence level. *Knowledge-Based Systems*, 108:110 – 124, 2016. ISSN 0950-7051. doi: http://dx.doi.org/10.1016/j.knosys.2016.05.040. URL `http://www.sciencedirect.com/science/article/pii/S095070511630137X`. New Avenues in Knowledge Bases for Natural Language Processing.

[14] Orestes Appel, Francisco Chiclana, Jenny Carter, and Hamido Fujita. *A consensus approach to sentiment analysis*, pages n/a–n/a. IEA/AIE 2017 : The 30th International Conference on Industrial, Engineering, Other Applications of Applied Intelligent Systems, Arras, France, 27-30 June 2017, Proceedings, 2017.

[15] Orestes Appel, Francisco Chiclana, Jenny Carter, and Hamido Fujita. *IOWA and Cross-ratio Uninorm operators as aggregation tools in sentiment analysis and ensemble methods (**Manuscript accepted: 11-March-2017**)*, pages n/a–n/a. IEEE: Proceedings of 2017 IEEE International Conference on Fuzzy Systems, Naples, Italy, 9-12 July 2017, Proceedings, 2017.

[16] Orestes Appel, Francisco Chiclana, Jenny Carter, and Hamido Fujita. *Successes and challenges in developing a hybrid approach to sentiment analysis (**Manuscript submitted: 08-Jan-2017**)*, page tba. Springer: Journal of Applied Intelligence. Special Edition S.I. : Knowledge-Based Systems and Data Sciences, Proceedings, 2017.

[17] Orestes Appel, Francisco Chiclana, Jenny Carter, and Hamido Fujita. A consensus approach to the sentiment analysis problem driven by support-based iowa majority. *International Journal of Intelligent Systems*, pages n/a–n/a, 2017. ISSN 1098-111X. doi: 10.1002/int.21878. URL http://dx.doi.org/10.1002/int.21878.

[18] Orestes Appel, Francisco Chiclana, Jenny Carter, and Hamido Fujita. Cross-ratio uninorms as an effective aggregation mechanism in Sentiment Analysis. *Knowledge-Based Systems*, pages n/a–n/a, 2017. doi: 10.1016/j.knosys.2017.02.08. URL http://dx.doi.org/10.1016/j.knosys.2017.02.028.

[19] AYLIEN. Naive Bayes for Dummies; A Simple Explanation. *Accessed on: 31 of January of 2017*, Jun 2015. URL http://blog.aylien.com/naive-bayes-for-dummies-a-simple-explanation/.

[20] B.D. Baets and J. Fodor. Van Melle's combining function in MYCIN is a representable uninorm: An alternative proof. *Fuzzy Sets Systems*, 104:133–136, 1999.

[21] Alexandra Balahur. *Methods and Resources for Sentiment Analysis in Multilingual Documents of Different Text Types*. PhD thesis, Department of Software and Computing Systems, University of Alicante (Universidad de Alicante), 2011.

[22] Carmen Banea, Rada Mihalcea, and Janyce Wiebe. A Bootstrapping Method for Building Subjectivity Lexicons for Languages with Scarce Resources. *In Proceedings of the International Conference on Language Resources and Evaluations (LREC 2008), Marrakech, Morocco*, pages 2764–2767, May 2008.

[23] Ann Banfield. *Unspeakable Sentences: Narration and Representation in the Language of Fiction*. Routledge and Kegan Paul, Law Book Co of Australasia, 1st edition, June 1982. ISBN 978-0710009050.

[24] Christian Becker-Asano. Seminar: Computational modeling of emotions. Held at the Isaac Ewton Institute for textscMathematical Ciences, Cambridge, UK; Tuesday 13 March 2012, 16:30-17:00, Seminar Room 1. *Albert-Ludwigs-Universität Freiburg, Computer Science Department, Foundation of Artificial Intelligence*, pages 1–17, 2012.

[25] Steven Bird. NLTK: the Natural Language Toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, COLING-ACL'06, pages 69–72, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. doi: 10.3115/1225403.1225421. URL http://dx.doi.org/10.3115/1225403.1225421.

[26] Steven Bird, Edward Loper, and Ewan Klein. *Natural Language Processing with Python*. O'Reilly Media Inc, 2009.

[27] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer Science + Business Media, LLC, 1st edition, 2006. ISBN 978-0-387-31073-2.

[28] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[29] William Frederick Blewitt. *Exploration of Emotion Modelling through Fuzzy Logic*. PhD thesis, De Montfort University, Great Britain, 2012.

[30] D. Bollegala and E. Shutova. Metaphor interpretation using paraphrases extracted from the web. *PLoS ONE e74304*, 8(9):1614–1617, Sep 2013. doi: 10.1371/journal.pone.0074304. URL http://www.plosone.org.

[31] Gloria Bordogna and Simone Sterlacchini. A multi criteria group decision making process based on the soft fusion of coherent evaluations of spatial alternatives. In Lotfi A. Zadeh, Ali M. Abbasov, Ronald R. Yager, Shahnaz N. Shahbazova, and Marek Z. Reformat, editors, *Recent Developments and New Directions in Soft Computing*, volume 317 of *Studies in Fuzziness and Soft Computing*, pages 65–79. Springer International Publishing, 2014. ISBN 978-3-319-06322-5. doi: 10.1007/978-3-319-06323-2_5. URL http://dx.doi.org/10.1007/978-3-319-06323-2_5.

[32] Jorge Luis Borges. *El libro de los seres imaginarios*. Biblioteca Borges, Alianza Editorial, Re-edited and augmented in 1967 & 1969 edition, 1957.

[33] Soheil Boroushaki and Jacek Malczewski. Using the fuzzy majority approach for gis-based multicriteria group decision-making. *Comput. Geosci.*, 36(3):302–312, Mar 2010. ISSN 0098-3004. doi: 10.1016/j.cageo.2009.05.011. URL http://dx.doi.org/10.1016/j.cageo.2009.05.011.

[34] Bernadette Bouchon-Meunier. *Aggregation and Fusion of Imperfect Information, Bernadette Bouchon-Meunier, Editor*, volume 12 of *Studies in Fuzziness and Soft Computing*. Springer-Verlag Berlin Heidelberg, 1998.

[35] Felipe Bravo-Márquez, Marcelo Mendoza, and Barbara Poblete. Meta-level sentiment models for big social data analysis. *Knowledge-Based Systems*, 69:86–99, 2014. doi: 10.1016/j.knosys.2014.05.016. URL http://dx.doi.org/10.1016/j.knosys.2014.05.016.

[36] Eric Brill. Some advances in transformation-based part of speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (Vol. 1)*, AAAI '94, pages 722–727, Menlo Park, CA, USA, 1994. American Association for Artificial Intelligence. ISBN 0-262-61102-3. URL http://dl.acm.org/citation.cfm?id=199288.199378.

[37] B. Buchanan and Editors E. Shortliffe. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley, Reading, MA, 1984.

[38] James J. Buckley and Esfandiar Eslami. *An Introduction to Fuzzy Logic and Fuzzy Sets*. Physica-Verlag, 1st edition, October 2002. ISBN 3-7908-1447-4.

[39] Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998. ISSN 1384-5810. doi: 10.1023/A:1009715923555. URL http://dx.doi.org/10.1023/A%3A1009715923555.

[40] Francisco Javier Cabrerizo, Francisco Chiclana, Rami Al-Hmouz, Ali Morfeq, Abdullah Saeed Balamash, and Enrique Herrera-Viedma. Fuzzy decision making and consensus: challenges. *Journal of Intelligent & Fuzzy Systems*, pages 1109–1118, 3 2015.

[41] Erik Cambria and Amir Hussain. *Sentic Computing: Techniques, Tools and Applications*. Springer: Springer Briefs in Cognitive Computation, first edition, 2012.

[42] Erik Cambria, Catherine Havasi, and Amir Hussain. Senticnet 2: A semantic and affective resource for opinion mining and sentiment analysis. In *Proceedings of the Twenty-Fifth International Florida Artificial Intelligence Research Society Conference, Marco Island, Florida. May 23-25, 2012*, 2012. URL http://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS12/paper/view/4411.

[43] Erik Cambria, Bjorn Schuller, Yunqing Xia, and Catherine Havasi. New Avenues in Opinion Mining and Sentiment Analysis. *IEEE Intelligent Systems*, 28(2):15–21, March 2013.

[44] Erik Cambria, Björn W. Schuller, Bing Liu, Haixun Wang, and Catherine Havasi. Knowledge-based approaches to concept-level sentiment analysis. *IEEE Intelligent Systems*, 28(2):12–14, 2013. doi: 10.1109/MIS.2013.45. URL http://dx.doi.org/10.1109/MIS.2013.45.

[45] Erik Cambria, Daniel Olsher, and Dheeraj Rajagopal. Senticnet 3: A common and common-sense knowledge base for cognition-driven sentiment analysis. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 1515–1521, 2014. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8479.

[46] Erik Cambria, Haixun Wang, and Bebo White. Guest editorial: Big social data analysis. *Knowledge-Based Systems*, 69:1 – 2, 2014. ISSN 0950-7051. doi: http://dx.doi.org/10.1016/j.knosys.2014.07.002. URL http://www.sciencedirect.com/science/article/pii/S0950705114002500.

[47] Jaime Carbonell. Metaphor: An Inescapable Phenomenon in Natural Language Comprehension. In Wendy Lehnert and Martin Ringle, editors, *Strategies for Natural Language Processing*, pages 415–434. Lawrence Erlbaum, 1982.

[48] Claire Cardie. Empirical methods in information extraction. *AI Magazine*, 18:65–79, 1997.

[49] Pimwadee Chaovalit and Lina Zhou. Movie Review Mining:a Comparison between Supervised and Unsupervised Classification Approaches. *Proceedings of the 38th Hawaii International Conference on System Sciences – 2005*, pages 1–9, 2005.

[50] Eugene Charniak. S*tatistical* L*anguage* L*earning*. The MIT Press, September 1996.

[51] Eugene Charniak. Statistical techniques for natural language parsing. *AI Magazine*, 18(4):33–44, August 1997.

[52] Francisco Chiclana and Shang-Ming Zhou. Type-reduction of general type-2 fuzzy sets: the type-1 OWA approach. *International Journal of Intelligent Systems*, 28(5):505–522, 2013.

[53] Francisco Chiclana, Francisco Herrera, and Enrique Herrera-Viedma. Integrating three representation models in fuzzy multipurpose decision making based on fuzzy preference relations. *Fuzzy Sets Syst.*, 97(1):33–48, Jul 1998. ISSN 0165-0114. doi: 10.1016/S0165-0114(96)00339-9. URL http://dx.doi.org/10.1016/S0165-0114(96)00339-9.

[54] Francisco Chiclana, Enrique Herrera-Viedma, Francisco Herrera, and Sergio Alonso. Some induced ordered weighted averaging operators and their use for solving group decision-making problems based on fuzzy preference relations. *European Journal of Operational Research*, 182(1):383–399, 2007.

[55] Francisco Chiclana, Enrique Herrera-Viedma, Sergio Alonso, and Francisco Herrera. Cardinal consistency of reciprocal preference relations: A characterization of multiplicative transitivity. *IEEE Transactions on Fuzzy Systems*, 17(1):14–23, 2009.

[56] Heeryon Cho, Songkuk Kim, Jongseo Lee, and Jong-Seok Lee. Data-driven integration of multiple sentiment dictionaries for lexicon-based sentiment classification of product reviews. *Knowledge-Based Systems*, 71:61–71, 2014. doi: 10.1016/j.knosys.2014.06.001. URL http://dx.doi.org/10.1016/j.knosys.2014.06.001.

[57] Noam Chomsky. *Syntactic Structures*. Mouton de Gruyter (formerly Mouton, The Hague), 2nd revised (2002) edition, 1957 1st edition. ISBN 3–11–017279–8.

[58] Noam Chomsky. *Aspects of the Theory of Syntax*. The MIT Press, 1st edition, March 1969. ISBN 0262530074.

[59] Kenneth Church, William Gale, Patrick Hanks, and Donald Hindle. Using statistics in lexical analysis. In *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon*, pages 115–164. Erlbaum, 1991.

[60] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. In *Proceedings of the 27th Annual Meeting on Association for Computational Linguistics*, ACL '89, pages 76–83, Stroudsburg, PA, USA, 1989. Association for Computational Linguistics. doi: 10.3115/981623.981633. URL http://dx.doi.org/10.3115/981623.981633.

[61] K.W. Church and P. Hanks. Word association norms, mutual information and lexicography. *Proceedings of the 27th Annual Conference of the ACL, New Brunswick, NJ: ACL*, pages 76–83, 1989.

[62] Martine De Cock and Etienne E. Kerre. Fuzzy modifiers based on fuzzy relations. *Inf. Sci. Inf. Comput. Sci.*, 160 (1-4):173–199, Mar 2004. ISSN 0020-0255. doi: 10.1016/j.ins.2003.09.002. URL http://dx.doi.org/10.1016/j.ins.2003.09.002.

[63] Martine De Cock, Ulrich Bodenhofer, and Etienne E. Kerre. Modelling linguistic expressions using fuzzy relations. In *Proceedings of the 6th International Conference on Soft Computing*, pages 353–360, October 2000.

[64] Anaïs Collomb, Crina Costea, Damien Joyeux, Omar Hasan, and Lionel Brunie. A study and comparison of sentiment analysis methods for reputation evaluation. *Research/Technical Report (RR-LIRIS-2014-002): Laboratoire d'InfoRmatique en Image et Systèmes d'information, UMR 5205 CNRS / INSA de Lyon / Université Claude Bernard Lyon 1 / Université Lumière Lyon 2 / École Centrale de Lyon, France*, 2014.

[65] Mita K. Dalal and Mukesh A. Zaveri. Semisupervised learning based opinion summarization and classification for online product reviews. *Applied Computational Intelligence and Soft Computing*, 2013(Article ID 910706), 2013. doi: http://dx.doi.org/10.1155/2013/910706.

[66] Mita K. Dalal and Mukesh A. Zaveri. Opinion mining from online user reviews using fuzzy linguistic hedges. *Appl. Comp. Intell. Soft Comput.*, 2014:1–9, Jan 2014.

[67] Robert Dale. *Classical Approaches to Natural Language Processing*. *In* Handbook of Natural Language Processing, Chapter 26, pp. 3–7, Chapman & Hall CRC, Eds: N. Indurkhya and F. J. Damerau, second edition, 2010.

[68] António Damásio. *Descartes' error: emotion, reason and the human brain*. Penguin Books, reprint edition (sept. 27 2005) edition, 1994. ISBN 0-399-13894-3,9978-0143036227.

[69] C. Darwin. *The expression of the emotions in man and animals*. New York: Filiquarian [published in 2007], 1872. ISBN 0-8014-1990-5.

[70] Sanjiv R. Das, Mike Y. Chen, To Vikas Agarwal, Chris Brooks, Yuk shee Chan, David Gibson, David Leinweber, Asis Martinez-Jerez, Priya Raghubir, Sridhar Rajagopalan, Ajit Ranade, Mark Rubinstein, and Peter Tufano. Yahoo! for Amazon: Sentiment extraction from small talk on the web. In *8th Asia Pacific Finance Association Annual Conference*, 2001.

[71] Xiaowen Ding, Bing Liu, and Philip S. Yu. A Holistic Lexicon Based Approach to Opinion Mining. *WSDM '08 Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 231–240, 2008.

[72] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10): 78–87, Oct 2012. ISSN 0001-0782. doi: 10.1145/2347736.2347755. URL http://doi.acm.org/10.1145/2347736.2347755.

[73] R. Kent Dybvig. *The Scheme Programming Language*. The MIT Press, 4th edition, 2009. ISBN 978-0-262-51298-5.

[74] Fabon Dzogang, Marie-Jeanne Lesot, Maria Rifqi, and Bernadette Bouchon-Meunier. Expressions of Graduality for Sentiments Analysis – A Survey. In *Fuzzy Systems (FUZZ), 2010 IEEE International Conference on*, pages 1–7, 2010.

[75] Albert Einstein. *Geometrie und Erfahrung*. See Einstein Archive, Berlin edition, 1921.

[76] P. Ekman(Ed.). *Emotion in the human face*. New York: Cambridge University Press, 1982.

[77] Magy Seif El-Nasr, John Yen, and Thomas R. Ioerger. Flame: Fuzzy logic adaptive model of emotions. *Autonomous Agents and Multi-Agent Systems*, 3(3):219–257, September 2000. ISSN 1387-2532. doi: 10.1023/A:1010030809960. URL http://dx.doi.org/10.1023/A:1010030809960.

[78] Andrea Esuli and Fabrizio Sebastiani. Determining the semantic orientation of terms through gloss classification. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM '05, pages 617–624, New York, NY, USA, 2005. ACM. ISBN 1-59593-140-6. doi: 10.1145/1099554.1099713. URL http://doi.acm.org/10.1145/1099554.1099713.

[79] Andrea Esuli and Fabrizio Sebastiani. SentiWordNet – A Publicly Available Lexical Resource for Opinion Mining. *In Proceedings of the 5th Conference on Language Resources and Evaluation (LREC06)*, pages 417–422, 2006.

[80] Andrea Esuli and Fabrizio Sebastiani. SentiWordNet: a high-coverage lexical resource for opinion mining. Technical Report ISTI-PP-002/2007, Institute of Information Science and Technologies (ISTI) of the Italian National Research Council (CNR), Oct 2006. URL http://tcc.itc.it/projects/ontotext/Publications/sentiWN-TR.pdf.

[81] Y. Lu et al. Automatic Construction of a Context-Aware Sentiment Lexicon:An Optimization Approach. *Proceedings 20th International Conf. World Wide Web (WWW), ACM*, pages 347–356, 2011.

[82] Ronen Feldman. Techniques and applications for sentiment analysis. *Communications of the ACM*, 56(4):82–89, April 2013.

[83] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. A Bradford Book. Series: Language, Speech and Communication. Christiane Fellenbaum (Editor), 1st. edition, 1998. ISBN 978-0262061971.

[84] J. R. Firth. A synopsis of linguistic theory 1930-55. *Studies in Linguistic Analysis (special volume of the Philological Society)*, 1952-59:1–32, 1957.

[85] Tristan Fletcher. *Tutorial Paper*: Support Vector Machines Explained. *University College London, Dept. of Computer Science, London, Great Britain*, March 2009. URL www.cs.ucl.ac.uk/staff/T.Fletcher/.

[86] János Fodor. On Rational Uninorms. *In Proceedings of the First Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence, Herlany, Slovakia, February 12-14, 2003*, pages 139–147, 2003.

[87] János Fodor. *Aggregation Functions in Fuzzy Systems, Fodor, János and Kacprzyk, Janusz, Editors*, pages 25–50. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-642-03633-0. doi: 10.1007/978-3-642-03633-0_2.

[88] Guohong Fu and Xin Wang. Chinese sentence-level sentiment classification based on fuzzy sets. In Chu-Ren Huang and Dan Jurafsky, editors, *COLING (Posters)*, pages 312–319. Chinese Information Processing Society of China, 2010. URL http://dblp.uni-trier.de/db/conf/coling/coling2010p.html#FuW10.

[89] Monalisa Ghosh and Animesh Kar. Unsupervised Linguistic Approach for Sentiment Classification from Online Reviews Using Sentiwordnet 3.0. *International Journal of Engineering Research & Technology (IJERT)*, 2(9):55–60, September 2013. ISSN 2278-0181.

[90] Siegfried Gottwald. *A Treatise on Many-Valued Logics*. Baldock, Hertfordshire, England: Research Studies Press Ltd., 1st. edition, 2001. ISBN 978-0-86380-262-1.

[91] Mohammad Sadegh Hajmohammadi, Roliana Ibrahim, Ali Selamat, and Hamido Fujita. Combination of active learning and self-training for cross-lingual sentiment classification with density analysis of unlabelled samples. *Information Sciences*, 317(C):67–77, Oct 2015. ISSN 0020-0255. doi: 10.1016/j.ins.2015.04.003. URL http://dx.doi.org/10.1016/j.ins.2015.04.003.

[92] V. Hatzivassiloglou and K.R. McKeown. Predicting the semantic orientation of adjectives. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL) and the 8th Conference of the European Chapter of the ACL. New Brunswick, NJ, USA: ACL*, pages 174–181, 1997.

[93] V. Hatzivassiloglou and J.M. Wiebe. Effects of adjective orientation and gradability on sentence subjectivity. In *Proceedings of the 18th International Conference on Computational Linguistics. New Brunswick, NJ, USA: ACL*, 2000.

[94] Vasileios Hatzivassiloglou and Kathleen McKeown. Towards the automatic identification of adjectival scales: Clustering adjectives according to meaning. In Lenhart K. Schubert, editor, *ACL: 31st Annual Meeting of the Association for Computational Linguistics (ACL), 22-26 June 1993, Ohio State University, Columbus, Ohio, USA, Proceedings*, pages 172–182. ACL, 1993.

[95] Marti A. Hearst and Xerox Palo Alto Research Center. Direction-based text interpretation as an information access refinement. *Text-Based Intelligent Systems*, pages 1–13, 1992.

[96] Bas Heerschop, Frank Goossen, Alexander Hogenboom, Flavius Frasincar, Uzay Kaymak, and Franciska de Jong. Polarity analysis of texts using discourse structure. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011*, pages 1061–1070, 2011. doi: 10.1145/2063576.2063730. URL http://doi.acm.org/10.1145/2063576.2063730.

[97] Francisco Herrera and Enrique Herrera-Viedma. Linguistic decision analysis: steps for solving decision problems under linguistic information. *Fuzzy Sets and Systems*, 115:67–82, 2000.

[98] Alexander Hogenboom, Daniella Bal, Flavius Frasincar, Malissa Bal, Franciska de Jong, and Uzay Kaymak. Exploiting emoticons in sentiment analysis. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, Coimbra, Portugal, March 18-22, 2013*, pages 703–710, 2013. doi: 10.1145/2480362.2480498. URL http://doi.acm.org/10.1145/2480362.2480498.

[99] Alexander Hogenboom, Daniella Bal, Flavius Frasincar, Malissa Bal, Franciska de Jong, and Uzay Kaymak. Exploiting Emoticons in Polarity Classification of Text. *Journal of Web Engineering*, 14(1&2):22–40, 2015. URL http://www.rintonpress.com/xjwe14/jwe-14-12/022-040.pdf.

[100] Alexander Hogenboom, Flavius Frasincar, Franciska de Jong, and Uzay Kaymak. Using rhetorical structure in sentiment analysis. *Communications of the ACM*, 58(7):69–77, 2015. doi: 10.1145/2699418. URL http://doi.acm.org/10.1145/2699418.

[101] John Hopcroft and J.D. Ullman. I*ntroduction to* A*utomata* T*heory,* L*anguages, and* C*omputation*. Addison Wesley, first edition, October 1979.

[102] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. *Proceedings – ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004 full paper), Seattle, Washington, USA, Aug. 22–25*, 2004.

[103] Sheng Huang, Zhendong Niu, and Chongyang Shi. Automatic construction of domain-specific sentiment lexicon based on constrained label propagation. *Knowledge-Based Systems*, 56:191–200, Jan 2014. ISSN 0950-7051. doi: 10.1016/j.knosys.2013.11.009. URL http://dx.doi.org/10.1016/j.knosys.2013.11.009.

[104] Eyke Hüllermeier. Fuzzy Methods in Machine Learning and Data Mining – Status and Prospects. *Fuzzy Sets and Systems (40th Anniversary of Fuzzy Sets)*, 156(3):387–406, December 2005.

[105] W. James. What is an emotion? *Mind*, 9:188–205, 1884.

[106] W. James. *The principles of psychology*. New York: Holt, 1890.

[107] Dragan Jočić and Ivana Štajner-Papuga. Distributivity equations and Mayor's aggregation operators. *Knowledge-Based Systems*, 52:194–200, Nov 2013.

[108] S. Jusoh and H.M. Alfawareh. Applying fuzzy sets for opinion mining. In *Computer Applications Technology (ICCAT), 2013 International Conference on*, pages 1–5, Jan 2013. doi: 10.1109/ICCAT.2013.6521965.

[109] Raquel Justo, Thomas Corcoran, Stephanie M. Lukin, Marilyn A. Walker, and M. Inés Torres. Extracting relevant knowledge for the detection of sarcasm and nastiness in the social web. *Knowledge-Based Systems*, 69:124–133, 2014. URL http://dblp.uni-trier.de/db/journals/kbs/kbs69.html#JustoCLWT14.

[110] J. Kacprzyk. Group decision making with a fuzzy linguistic majority. *Fuzzy Sets and Systems*, 18:105–118, 1986.

[111] Jaap Kamps, Maarten Marx, Robert J. Mokken, and Maarten de Rijke. Using WordNet to measure semantic orientations of adjectives. In *Proceedings of LREC-04, 4th International Conference on Language Resources and Evaluation*, volume IV of *LREC '04*, pages 1115–1118, 2004.

[112] V. R. Kanagavalli and K. Raja. Detecting and resolving spatial ambiguity in text using named entity extraction and self learning fuzzy logic techniques. *CoRR*, abs/1303.0445, 2013.

[113] Animesh Kar and Deba Prasad Mandal. Finding Opinion Strength Using Fuzzy Logic on Web Reviews. *International Journal of Engineering and Industries*, 2(1), March 2011.

[114] Jayashri Khairnar and Mayura Kinikar. Machine Learning Algorithms for Opinion Mining and Sentiment Classification. *International Journal of Scientific and Research Publications*, 3(6):1–6, June 2013.

[115] Aurangzeb Khan, Baharum Baharudin, and Khairullah Khan. Sentiment classification from online customer reviews using lexical contextual sentence structure. In *Software Engineering and Computer Systems*, pages 317–331. Springer Berlin Heidelberg, 2011.

[116] R. Khoury, F. Karray, Yu Sun, M. Kamel, and O. Basir. Semantic understanding of general linguistic items by means of fuzzy set theory. *IEEE Transactions on Fuzzy Systems*, 15(5):757–771, Oct 2007. ISSN 1063-6706.

[117] E.P. Klement, R. Mesiar, and E. Pap. On the relationship of associative compensatory operators to triangular norms and conorms. *Int. J. Uncertainty, Fuzziness Knoledge-Based Systems*, 4:129–144, 1996.

[118] George Klir and Bo Yuan. *Fuzzy Sets and Fuzzy Logic, Theory and Applications*. New Jersey, USA: Prentice Hall PTR, first edition, 1995.

[119] George J. Klir, Ute H. St. Clair, and Bo Yuan. *Fuzzy Set Theory: Foundations and Applicationss*. New Jersey, USA: Prentice Hall PTR, first edition, 1997. ISBN 0-13-341058-7.

[120] Arnd Christian König and Eric Brill. Reducing the human overhead in text categorization. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 598–603, New York, NY, USA, 2006. ACM. ISBN 1-59593-339-5. doi: 10.1145/1150402.1150474. URL http://doi.acm.org/10.1145/1150402.1150474.

[121] Igor Kononenko and Matjaž Kukar. *Machine Learning and Data Mining: Introduction to Principles and Algorithms*. Horwood Publishing Limited, 2007. ISBN 1904275214, 9781904275213.

[122] Rudolf Kruse, Detlef Nauck, and Christian Borgelt. Data Mining with Fuzzy Methods – Status and Perspectives. *In Proceedings of the EUFIT99 – European Laboratory for Intelligent Techniques Engineering*, September 1999.

[123] Akshi Kumar and Teeja Mary Sebastian. Sentiment analysis: A perspective on its past, present and future. *International Journal of Intelligent Systems and Applications (IJISA), ISSN: 2074-904X (Print), ISSN: 2074-9058 (Online)*, 4(10):1–14, September 2012.

[124] Akshi Kumar and Teeja Mary Sebastian. Machine Learning assisted Sentiment Analysis. *Proceedings of International Conference on Computer Science and Engineering (ICCSE 2012)*, pages 123–130, 2012.

[125] D. Terence Langendoen. Studies in linguistic analysis. *Language (Published by: Linguistic Society of America)*, 40(2):305–321, Apr-Jun 1964. ISSN 1046-8188. URL http://www.jstor.org/stable/411592.

[126] R.Y. Lau, C. Lai, and Y. Li. Leveraging the Web for Context-Sensitive Opinion Mining. *Proceedings of 2nd IEEE International Conf. Computer Science and Information Technology (ICCSIT), IEEE*, pages 467–471, 2009.

[127] Teresa León, Nuria Ramón, José L. Ruiz, and Inmaculada Sirvent. Using induced ordered weighted averaging (iowa) operators for aggregation in cross-efficiency evaluations. *International Journal of Intelligent Systems*, 29 (12):1100–1116, 2014. ISSN 1098-111X. doi: 10.1002/int.21685. URL http://dx.doi.org/10.1002/int.21685.

[128] Chenghua Lin, Yulan He, Richard Everson, and Stefan Rüger. Weakly supervised joint sentiment-topic detection from text. *IEEE Transactions on Knowledge and Data Engineering*, 24(6):1134–1145, June 2012. ISSN 1041-4347. doi: 10.1109/TKDE.2011.48.

[129] Bing Liu. *Sentiment* A*nalysis and* S*ubjectivity*. *In* Handbook of Natural Language Processing, Chapter 26, pp. 627–666, Chapman & Hall CRC, Eds: N. Indurkhya and F. J. Damerau, second edition, 2010.

[130] Bing Liu. Sentiment analysis: A multifaceted problem. *IEEE Intelligent Systems*, 25(3):76–80, May 2010.

[131] Bing Liu. *Web Data Mining - Exploring Hyperlinks, Contents, and Usage Data*. Springer-Verlag Berlin Heidelberg, second edition, 2011. ISBN 978-3-642-19459-7.

[132] Bing Liu. *Tutorial*: Sentiment Analysis Tutorial Given at AAAI-2011, Monday, August 8 2011, San Francisco, USA. *Twenty-*F*ifth* C*onference on* A*rtificial* I*ntelligence (*AAAI*-11), held in* S*an* F*rancisco,* C*alifornia at the* H*yatt* R*egency* S*an* F*rancisco*, August 7–11 2011.

[133] Bing Liu. *Sentiment Analysis and Opinion Mining*. Morgan and Claypool Publishers: Synthesis Lectures on Human Language Technologies, 1st edition, 2012.

[134] Bin Lu and Benjamin K. Tsou. Combining a large sentiment lexicon and machine learning for subjectivity classification. In *Proceedings of the Ninth IEEE International Conference on Machine Learning and Cybernetics, Qingdao, 11–14 July, 2010*, pages 3311–3316, 2010.

[135] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 142–150, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-87-9. URL http://dl.acm.org/citation.cfm?id=2002472.2002491.

[136] Isa Maks and Piek Vossen. A verb lexicon model for deep sentiment analysis and opinion mining applications. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis (WASSA 2011)*, pages 10–18, Portland, Oregon, Jun 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-1702.

[137] R. L. Mandryk and M. S. Atkins. A fuzzy physiological approach for continuously modeling emotions during interaction with play technologies. *International Journal of Human-Computer Studies*, 65:329–347, 2007.

[138] W.C Mann and S.A. Thompson. Rhetorical Structure Theory: Toward a functional theory of text organization. *Text*, 8(3):243–281, October 1988.

[139] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, reprinted 2008, 2009 edition, 2008. ISBN 978–0–521–86571–5.

[140] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

[141] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330, Jun 1993. ISSN 0891-2017. URL http://dl.acm.org/citation.cfm?id=972470.972475.

[142] Stephen Marsland. *Machine Learning - An Algorithmic Perspective*. Chapman & Hall/CRC Machine Learning & Pattern Recognition Series CRC Press, 1st edition, 2009. ISBN 978-1-4200-6718-7 (Hardcover).

[143] Francisco Mata, Luis G. Pérez, Shang-Ming Zhou, and Francisco Chiclana. Type-1 OWA methodology to consensus reaching process in multi-granular linguistic contexts. *Knowledge-Based Systems*, 58:11–22, 2014.

[144] Francisco Mata, Luis G. Pérez, Francisco Chiclana, and Enrique Herrera-Viedma. Aggregation on Unbalanced Fuzzy Linguistic Information in Decision Problems based on Type-1 OWA Operator. *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE2015), Istanbul, August 2-5, 2015*, pages 1–6, August 2015. doi: 10.1109/FUZZ-IEEE.2015.7337995.

[145] John McCarthy. Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I. *Communications of the ACM*, 3(4):184–195, Apr 1960. ISSN 0001-0782. doi: 10.1145/367177.367199. URL http://doi.acm.org/10.1145/367177.367199.

[146] Fanyong Meng and Xiaohong Chen. A new method for group decision making with incomplete fuzzy preference relations. *Knowledge-Based Systems*, 73:111–123, Jan 2015.

[147] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. Introduction to WordNet: an on-line lexical database. *International Journal of Lexicography*, 3(4):235–244, 1990. URL http://wordnetcode.princeton.edu/5papers.pdf.

[148] G.A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63:81–97, 1956.

[149] George Miller and Walter Charles. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28, 1991.

[150] H. B. Mitchell and D. D. Estrakh. A modified OWA operator and its use in lossless DPCM image compression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, pages 429–436, 1997.

[151] Tom Mitchell. *Machine Learning*. McGraw Hill Higher Education, New edition edition, October 1997.

[152] Samaneh Moghaddam and Martin Ester. Ilda: Interdependent lda model for learning latent aspects and their ratings from online product reviews. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 665–674, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0757-4. doi: 10.1145/2009916.2010006. URL http://doi.acm.org/10.1145/2009916.2010006.

[153] Rick L. Morgan and David Heise. Structure of Emotions. *Social Psychology Quarterly*, 51(1):19–31, 1988.

[154] Andrius Mudinas, Dell Zhang, and Mark Levene. Combining lexicon and learning based approaches for concept-level sentiment analysis. In *Proceedings of the First International Workshop on Issues of Sentiment Discovery and Opinion Mining*, WISDOM '12, pages 51–58, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1543-2. doi: 10.1145/2346676.2346681. URL http://doi.acm.org/10.1145/2346676.2346681.

[155] Samaneh Nadali, Masrah Murad, and Rabiah Kadir. Sentiment classification of customer reviews based on fuzzy logic. *In Information Technology (ITSim), 2010 International Symposium in (Volume:2), Kuala Lumpur, Malaysia*, 2:1037–1040, June 2010.

[156] Dana Nau. Part-of-speech tagging (pdf slides). *CMSC 421, Introduction to AI, Spring 2010, Lecture Notes, University of Maryland, Computer Science Department*, Spring 2010. URL http://www.cs.umd.edu/~nau/cmsc421/part-of-speech-tagging.pdf.

[157] Hwee Tou Ng and John Zelle. Corpus-based approaches to semantic interpretation in nlp. *AI Magazine*, 18(4): 45–64, August 1997.

[158] Kamal Nigam, John Lafferty, and Andrew McCallum. Using maximum entropy for text classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering*, volume 1, pages 61–67, 1999.

[159] M. Ochs, R. Niewiadomski, and C. P. D. Sadek. Intelligent Expressions of Emotions. *In Affective Computing and Intelligent Interaction, S. B. Heidelberg, Ed.*, 3784:707–714, 2005.

[160] A. Ortony, G. L. Clore, and M. A. Foss. The psychological foundations of the affective lexicon. *Journal of Personality and Social Psychology*, 53:751–766, 1987.

[161] Andrew Ortony, Gerald L. Clore, and Allan Collins. *The Cognitive Structure of Emotions*. Cambridge University Press, first edition, 1988.

[162] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, (ACL '04)*, pages 271–278, 2004.

[163] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL '05)*, ACL'05, pages 115–124, 2005.

[164] Bo Pang and Lillian Lee. *Opinion mining and sentiment analysis*. NOW: the essence of knowledge, Foundations and Trends in Information Retrieval, Vol. 2, Nos. 1–2, pp. 1–135, 2008.

[165] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? Sentiment Classification using Machine Learning Techniques. *Proceedings of the Association for Computational Linguistics (ACL-02) Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 10:79–86, 2002.

[166] Patrick Pantel and Dekang Lin. Discovering word senses from text. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 613–619, New York, NY, USA, 2002. ACM. ISBN 1-58113-567-X. doi: 10.1145/775047.775138. URL http://doi.acm.org/10.1145/775047.775138.

[167] Brian Parkinson, Agneta H. Fisher, and Antony S. R. Manstead. *Emotion in Social Relations: Cultural, Group, and Interpersonal Processes*. Psychology Press, new ed edition (november 26, 2004) edition, 2005. ISBN 978-1-841-69046-9.

[168] Gabriella Pasi and Ronald R. Yager. Modeling the concept of majority opinion in group decision making. *Inf. Sci.*, 176(4):390–414, Feb 2006. ISSN 0020-0255. doi: 10.1016/j.ins.2005.07.006. URL http://dx.doi.org/10.1016/j.ins.2005.07.006.

[169] J.I. Peláez, J. M. Doña, and J.A. Gómez-Ruiz. Analysis of OWA operators in decision making for modelling the majority concept. *Applied Mathematics and Computation*, 186:1263–1275, 2007. doi: 10.1016/j.amc.2006.07.161.

[170] Luis G. Pérez, Francisco Mata, and Francisco Chiclana. Social Network Decision Making with Linguistic Trustworthiness-Based Induced OWA Operators. *International Journal of Intelligent Systems*, 29(12):1117–1137, 2014. ISSN 1098-111X. doi: 10.1002/int.21686. URL http://dx.doi.org/10.1002/int.21686.

[171] Luis G. Pérez, Francisco Mata, Francisco Chiclana, Gang Kou, and Enrique Herrera-Viedma. Modeling influence in group decision making. *Soft Computing*, 20(4):1653–1665, 2016. doi: 10.1007/s00500-015-2002-0. URL http://dx.doi.org/10.1007/s00500-015-2002-0.

[172] Patrizia Pérez-Asurmendi and Francisco Chiclana. Linguistic majorities with difference in support. *Applied Soft Computing*, 18:196–208, 2004.

[173] Jacob Perkins. *Python Text Processing with NLTK 2.0 Cookbook*. Packt Publishing, 2010. ISBN 978-1-84951-360-9.

[174] R. Plutchik. *The Emotions: Facts, Theories, and a new model*. Mouton de Gruyter (formerly Mouton, The Hague), 1962.

[175] R. Plutchik. Emotion: Theory, research, and experience. *Theories of emotion. Chapter: A general psychoevolutionary theory of emotion. New York: Academic*, 1:3–33, 1980.

[176] Soujanya Poria, Erik Cambria, Grégoire Winterstein, and Guang-Bin Huang. Sentic patterns: Dependency-based rules for concept-level sentiment analysis. *Knowledge-Based Systems*, 69:45–63, 2014. doi: 10.1016/j.knosys.2014.05.005. URL http://dx.doi.org/10.1016/j.knosys.2014.05.005.

[177] Chris Potts. Sentiment Symposium Tutorial: Linguistic structure (part of the Sentiment Analysis Symposium held at San Francisco, november 8-9, 2011). Stanford Department of Linguistics, Stanford University. Accessed date: December 2014, November 2011. URL http://sentiment.christopherpotts.net/index.html.

[178] Chris Potts. CS 224U / LING 188/288: Natural Language Understanding Class. Homework Data and Python Code for VSM. Stanford Department of Linguistics, Stanford University. Accessed date: December 2014, Spring 2014. URL http://web.stanford.edu/class/cs224u/.

[179] Christopher Potts. Sentiment Symposium Tutorial. *Sentiment Analysis Symposium, held in* San Francisco, California, *November 8-9*, 2011.

[180] Gang Qian and Ze-Shui Xu. Extended IOWA operator and its application to group decision making with linguistic preference information. *Proceedings of the Fifth International Conference on Machine Learning and Cybernetics, Dalian, 13-16 August 2006*, pages 1662–1666, August 2006.

[181] Colorado Reed. Tutorial: Latent dirichlet allocation: Towards a deeper understanding. *Unpublished: PhD Student at University of California, Berkeley*, pages 1–13, January 2012. URL http://obphio.us/writings/.

[182] Vassiliki Rentoumi, George A. Vouros, Vangelis Karkaletsis, and Amalia Moser. Investigating metaphorical language in sentiment analysis: A sense-to-sentiment perspective. *ACM Trans. Speech Lang. Process.*, 9(3):6:1–6:31, Nov 2012. ISSN 1550-4875. doi: 10.1145/2382434.2382436. URL http://doi.acm.org/10.1145/2382434.2382436.

[183] Brian Roark and Richard Sproat. *Computational Approaches to Morphology and Syntax*. Oxford University Press - Oxford Linguistics, 1st edition, 2007. ISBN 978-0-19-927478-9.

[184] Timothy J. Ross. *Fuzzy Logic with Engineering Applications*. John Wiley & Sons, Ltd., 3rd edition, 2010. ISBN 978-0-470-74376-8.

[185] John Rothfels and Julie Tibshirani. *Report*: Unsupervised sentiment classification of english movie reviews using automatic selection of positive and negative sentiment items. *Stanford University, CS224n-Ling237 NLP Reports*, 2010. URL http://nlp.stanford.edu/courses/cs224n/2010/reports/rothfels-jtibs.pdf.

[186] Imre J. Rudas and János Fodor. Information Aggregation in Intelligent Systems Using Generalized Operators. *International Journal of Computers, Communications & Control*, I(1):47–57, 2006.

[187] Imre J. Rudas, Endre Pap, and János Fodor. Editorial: Special issue on Advances in fuzzy knowledge systems: Theory and application. *Knowledge-Based Systems*, 38:1–2, Jan 2013.

[188] Imre J. Rudas, Endre Pap, and János Fodor. Information aggregation in intelligent systems: An application oriented approach. *Knowledge-Based Systems*, 38:3–13, Jan 2013.

[189] Mohammad Sadegh and Roliana Ibrahim Zulaiha Ali Othman. Combining Lexicon-based and Learning-based Methods for Twitter Sentiment Analysis. *International Journal of Computers & Technology*, 2(3):171–178, June 2012. ISSN 2277-3061 (online).

[190] G. Salton, A. A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, Nov 1975. ISSN 0001-0782. doi: 10.1145/361219.361220. URL http://doi.acm.org/10.1145/361219.361220.

[191] Gerard Salton. *The* SMART *Retrieval System - Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971.

[192] Beatrice Santorini. Part-of-speech tagging guidelines for the penn treebank project (3rd revision, 2nd printing). *Technical Report, Department of Computer and Information Science, University of Pennsylvania.*, 1995.

[193] Tyler Schnoebelen. Reading notes: Ortony et al. the cognitive structure of emotions (1988). *Tyler Schnoebelen's Academic Site: Language, Design, Linguistics*, 2009. URL http://web.stanford.edu/~tylers/emotions.shtml.

[194] Kim Schouten and Flavius Frasincar. Survey on aspect-level sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering*, 28(3):813–830, 2016. doi: 10.1109/TKDE.2015.2485209. URL http://dx.doi.org/10.1109/TKDE.2015.2485209.

[195] Ekaterina Shutova. Models of metaphor in NLP. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 688–697, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1858681.1858752.

[196] Ekaterina Shutova. Automatic metaphor interpretation as a paraphrasing task. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 1029–1037, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. ISBN 1-932432-65-5. URL http://dl.acm.org/citation.cfm?id=1857999.1858145.

[197] Ekaterina Shutova, Lin Sun, and Anna Korhonen. Metaphor identification using verb and noun clustering. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 1002–1010, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1873781.1873894.

[198] R. Socher, A. Perelygin, J.Y. Wu, J. Chuang, C.D. Manning, A.Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1631, 2013.

[199] Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. Semantic Compositionality through Recursive Matrix-Vector Spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 1201–1211, 2012. URL http://www.aclweb.org/anthology/D12-1110.

[200] R. Srivastaval, MPS Bhatia, H.K. Srivastava, and C.P. Sahu. Effects of adjective orientation and gradability on sentence subjectivity. *In IEEE International Conference on Computer & Communication Technology (ICCCT'10), 17–19 September 2010, Allahabad, Uttar Pradesh.*, pages 768–775, 2010.

[201] Bas R. Steunebrink. *The Logical Structure of Emotions*. PhD thesis, Utrecht University, The Netherlands, 2010.

[202] Bas R. Steunebrink, Mehdi Dastani, and John-Jules Ch. Meyer. The OCC Model Revisited. In *Proceedings of the 4th Workshop on Emotion and Computing - Current Research and Future Impact. Paderborn, Germany*, 2009.

[203] Ladda Suanmali, Naomie Salim, and Mohammed Salem Binwahlan. Fuzzy logic based method for improving text summarization. *International Journal of Computer Science and Information Security (IJCSIS)*, 2(1), June 2009.

[204] Pero Subasic and Alison Huettner. Affect Analysis of Text Using Fuzzy Semantic Typing. *Presented at the Proc. of FUZZ-IEEE 2000, The 9th International Conference on Fuzzy Systems, San Antonio, Texas*, 2000.

[205] Pero Subasic and Alison Huettner. Affect Analysis of Text Using Fuzzy Semantic Typing. *IEEE Transactions on Fuzzy Systems*, 9(4):483–496, August 2001.

[206] V. S. Subrahmanian and Diego Reforgiato Recupero. AVA: Adjective-Verb-Adverb Combinations for Sentiment Analysis. *IEEE Intelligent Systems*, 23(4):43–50, 2008.

[207] Gerald Jay Sussman and Guy Lewis Steele Jr. Scheme: An Interpreter for Extended Lambda Calculus. *MIT AI Lab. AI Lab Memo AIM-349*, December 1975.

[208] Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. Lexicon-Based Methods for Sentiment Analysis. *Computational Linguistics*, 37(2):267–307, 2011.

[209] Huifeng Tang, Songbo Tan, and Xueqi Cheng. A survey on sentiment detection of reviews. *Expert Systems with Applications (ELSEVIER)*, 36:10760–10773, 2009.

[210] Peter D. Turney. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the 12th European Conference on Machine Learning*, EMCL '01, pages 491–502, London, UK, UK, 2001. Springer-Verlag. ISBN 3-540-42536-5. URL http://dl.acm.org/citation.cfm?id=645328.650004.

[211] Peter D. Turney. Mining the Web for Synonyms: PMI-IR Versus LSA on TOEFL. In *Proceedings of the 12th European Conference on Machine Learning*, EMCL '01, pages 491–502, London, UK, UK, 2001. Springer-Verlag. ISBN 3-540-42536-5. URL http://dl.acm.org/citation.cfm?id=645328.650004.

[212] Peter D. Turney. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification Reviews. *Proceedings of the 40th* A*nnual* M*eeting of the* A*ssociation for* C*omputational* L*inguistics (*ACL*),* P*hiladelphia,* USA, pages 417–424, July 2002.

[213] Peter D. Turney and Michael L. Littman. Unsupervised learning of semantic orientation from a hundred-billion-word corpus. *CoRR*, cs.LG/0212012, 2002.

[214] Peter D. Turney and Michael L. Littman. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Trans. Inf. Syst.*, 21(4):315–346, Oct 2003. ISSN 1046-8188. doi: 10.1145/944012.944013. URL http://doi.acm.org/10.1145/944012.944013.

[215] Peter D. Turney and Michael L. Littman. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Trans. Inf. Syst.*, 21(4):315–346, Oct 2003. ISSN 1046-8188. doi: 10.1145/944012.944013. URL http://doi.acm.org/10.1145/944012.944013.

[216] Peter D. Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *J. Artif. Int. Res.*, 37(1):141–188, Jan 2010. ISSN 1076-9757. URL http://dl.acm.org/citation.cfm?id=1861751.1861756.

[217] Raquel Ureña, Francisco Chiclana, Hamido Fujita, and Enrique Herrera-Viedma. Confidence-consistency driven group decision making approach with incomplete reciprocal intuitionistic preference relations. *Knowledge-Based Systems*, 89:86–96, Nov 2015.

[218] M.S. Usha and M. Indra Devi. Analysis of sentiments using unsupervised learning techniques. In *Information Communication and Embedded Systems (ICICES), 2013 International Conference on*, pages 241–245, Feb 2013. doi: 10.1109/ICICES.2013.6508203.

[219] Albert van der Heide, Daniel Sánchez, and Gracián Triviño. Computational models of affect and fuzzy logic. In Sylvie Galichet, Javier Montero, and Gilles Mauris, editors, *Proceedings of the 7th conference of the European Society for Fuzzy Logic and Technology, EUSFLAT 2011, Aix-Les-Bains, France, July 18-22, 2011*, pages 620–627. Atlantis Press, 2011. ISBN 978-90-78677-00-0.

[220] Aline A. Vanin, Larissa A. de Freitas, , Renata Vieira, and Marco N. Bochernitsan. Some clues on irony detection in tweets. In *Proceedings of the 22Nd International Conference on World Wide Web Companion*, WWW '13 Companion, pages 635–636, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-2038-2. URL http://dl.acm.org/citation.cfm?id=2487788.2488012.

[221] Vladimir Vapnik. *Estimation of Dependencies Based on Empirical Data [in Russian]. Nauka, Moscow, 1979*. English translation: Springer-Verlag, New York, USA, 1982, 1982.

[222] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, USA, 1995.

[223] Vladimir Vapnik. *Statistical Learning Theory*. John Wiley and Sons, Inc., New York, USA, 1998.

[224] G. Vinodhini and RM. Chandrasekaran. Sentiment Analysis and Opinion Mining: A Survey. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(6):282–292, June 2012.

[225] Sida Wang and Christopher D. Manning. Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012): Short Papers*, 2:90–94, 2012.

[226] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 2009.

[227] Chunfu Wei and Ruifu Yuan. Decision-making Method based on Linguistic Aggregation Operators for Coal Mine Safety Evaluation. In *2010 International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, pages 17–20, Nov 2010. doi: 10.1109/ISKE.2010.5680786.

[228] Chunfu Wei, Zheng Pei, and Bo Li. Multiple Attribute Decision Making Based on Induced OWA operator. *Proceedings of the 2009 IEEE International Conference on Fuzzy Systems (FUZZ- IEEE2009), Jeju Island, Korea, August 20-24, 2009*, pages 1763–1766, August 2009.

[229] Wei Wei. Analyzing text data for opinion mining. In *Proceedings of the 16th International Conference on Natural Language Processing and Information Systems*, NLDB'11, pages 330–335, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-22326-6. URL http://dl.acm.org/citation.cfm?id=2026011.2026064.

[230] Albert Weichselbraun, Stefan Gindl, and Arno Scharl. Extracting and Grounding Contextualized Sentiment Lexicons. *IEEE Intelligent Systems*, 28(2):39–46, March 2013.

[231] Dominic Widdows. *Geometry and Meaning*. CSLI Publications, CSLI Lecture Notes No. 172, 1st edition, 2004. ISBN 1-57586-448-7.

[232] Janyce Wiebe. Identifying subjective characters in narrative. In *Proceedings of the International Conference on Computational Linguistics*, COLING '90. Association for Computational Linguistics, 1990.

[233] Janyce Wiebe. Tracking point of view in narrative. *Comput. Linguist.*, 20(2):233–287, Jun 1994. ISSN 0891-2017. URL http://dl.acm.org/citation.cfm?id=972525.972529.

[234] Janyce Wiebe. Learning subjective adjectives from corpora. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 735–740. AAAI Press, 2000. ISBN 0-262-51112-6. URL http://dl.acm.org/citation.cfm?id=647288.721121.

[235] Janyce Wiebe and Ellen Riloff. Finding mutual benefit between subjectivity analysis and information extraction. *IEEE Transactions on Affective Computing*, 2(4):175–191, 2011. ISSN 1949-3045. doi: http://doi.ieeecomputersociety.org/10.1109/T-AFFC.2011.19.

[236] Yorick Wilks and Janusz Bien. Beliefs, points of view, and multiple environments. *Cognitive Science*, 7(2): 95–119, 1983. ISSN 0364-0213. doi: http://dx.doi.org/10.1016/S0364-0213(83)80007-X. URL http://www.sciencedirect.com/science/article/pii/S036402138380007X.

[237] T. Wilson, J. Wiebe, and P. Hoffmann. Recognizing Contextual Polarity: An Exploration of Features for Phrase-Level Sentiment Analysis. *Computational Linguistics*, 35(3):399–433, 2009.

[238] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 347–354, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1220575.1220619. URL http://dx.doi.org/10.3115/1220575.1220619.

[239] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis. *Comput. Linguist.*, 35(3):399–433, September 2009. ISSN 0891-2017. doi: 10.1162/coli.08-012-R1-06-90. URL http://dx.doi.org/10.1162/coli.08-012-R1-06-90.

[240] T. Winograd. *Language as a Cognitive Process, Volume I: Syntax*. Addison-Wesley, Reading, MA, USA, 1983.

[241] R. Witte and S. Bergler. Fuzzy coreference resolution for summarization. *In proceedings of 2003 International Symposium on Reference Resolution and Its Applications to Question Answering and Summarization (ARQAS). Venice, Italy: Universita Ca' Foscari*, pages 43–50, 2003.

[242] Jian Wu and Francisco Chiclana. A social network analysis trust-consensus based approach to group decision-making problems with interval-valued fuzzy reciprocal preference relations. *Knowledge-Based Systems*, 59:97–107, 2014.

[243] Jian Wu and Francisco Chiclana. Multiplicative consistency of intuitionistic reciprocal preference relations and its application to missing values estimation and consensus building. *Knowledge-Based Systems*, 71:187–200, Nov 2014.

[244] Jian Wu and Francisco Chiclana. Trust based consensus model for social network in an incomplete linguistic information context. *Applied Soft Computing*, 35:827–839, 2015.

[245] Jian Wu, Francisco Chiclana, and Enrique Herrera-Viedma. Trust based consensus model for social network in an incomplete linguistic information context. *Applied Soft Computing*, pages 827–839, 2015.

[246] Jian Wu, Francisco Chiclana, and Huchang Liao. Isomorphic multiplicative transitivity for intuitionistic and interval-valued fuzzy preference relations and its application in deriving their priority vectors. *IEEE Transactions on Fuzzy Systems*, 2016. doi: 10.1109/TFUZZ.2016.2646749.

[247] Jian Wu, Ruoyun Xiong, and Francisco Chiclana. Uninorm trust propagation and aggregation methods for group decision making in social network with four tuples information. *Knowledge-Based Systems*, 96:29–39, 2016.

[248] Yusheng Xie, Zhengzhang Chen, Kunpeng Zhang, Yu Cheng, Daniel K. Honbo, Ankit Agrawal, and Alok N. Choudhary. MuSES: a multilingual sentiment elicitation system for Social Media Data. *IEEE Intelligent Systems*, 29(4):34–42, July 2014. ISSN 1541-1672.

[249] Huong Nguyen Thi Xuan, Anh Cuong Le, and Le Minh Nguyen. Linguistic features for subjectivity classification. In *Asian Language Processing (IALP), 2012 International Conference on*, pages 17–20, Nov 2012. doi: 10.1109/IALP.2012.47.

[250] Ronald R. Yager. Families of OWA Operators. *Fuzzy Sets Syst.*, 59(2):125–148, Oct 1993. ISSN 0165-0114. doi: 10.1016/0165-0114(93)90194-M. URL http://dx.doi.org/10.1016/0165-0114(93)90194-M.

[251] Ronald R. Yager. Quantifier guided aggregation using OWA operators. *International Journal of Intelligent Systems*, 11(1):49–73, 1996. ISSN 1098-111X. doi: 10.1002/(SICI)1098-111X(199601)11:1⟨49::AID-INT3⟩3.0.CO;2-Z. URL http://dx.doi.org/10.1002/(SICI)1098-111X(199601)11:1<49::AID-INT3>3.0.CO;2-Z.

[252] Ronald R. Yager. Induced aggregation operators. *Fuzzy Sets and Systems*, pages 59–69, 2003.

[253] Ronald R. Yager and Naif Alajlan. An intelligent interactive approach to group aggregation of subjective probabilities. *Knowledge-Based Systems*, 83:170–175, July 2015.

[254] Ronald R. Yager and Naif Alajlan. Some issues on the owa aggregation with importance weighted arguments. *Knowledge-Based Systems*, 100:89–96, May 2016.

[255] Ronald R. Yager and Alexander Rybalov. Uninorm aggregation operators. *Fuzzy Sets and Systems*, 80(1):111 – 120, 1996. ISSN 0165-0114. Fuzzy Modeling.

[256] R.R. Yager. On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decision Making. *IEEE Transactions on Systems Man and Cybernetics*, 18(1):183–190, Jan 1988. ISSN 0018-9472. doi: 10.1109/21.87068.

[257] R.R Yager. The power average operator. *Transactions on Systems, Man, and Cybernetics, Part A: Cybernetics*, 31: 724–730, 2001.

[258] R.R Yager and D.P. Filev. Induced ordered weighted averaging operators. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, 29(2):141–150, Apr 1999. ISSN 1083-4419. doi: 10.1109/3477.752789.

[259] L.A. Zadeh. Similarity Relations and Fuzzy Orderings. *Information Sciences*, 3(2):177–200, Apr 1971. ISSN 0020-0255. doi: 10.1016/S0020-0255(71)80005-1. URL `http://dx.doi.org/10.1016/S0020-0255(71)80005-1`.

[260] L.A. Zadeh. A computational approach to fuzzy qualifiers in natural languages. *Computing and Mathematics with Applications*, 9(1):149–184, 1983.

[261] Lotfi A. Zadeh. Fuzzy sets. *Information and Control Proceedings of the Annual Conference of the Association for the Advancement of Artifical Intelligence (AAAI)*, 8:338–353, 1965.

[262] Lotfi A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man and Cybernetics, Part B-Cybernetics*, SMC-3:28–44, 1973.

[263] Lotfi A. Zadeh. The concept of a linguistic variable and its applications to approximate reasoning - I. *Information Sciences*, 8:199–249, 1975.

[264] Lotfi A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1:3–28, 1978.

[265] Lotfi A. Zadeh. Fuzzy probabilities. *Information Processing and Management - Special Issue Information Theory Applications to Problems of Information Science*, 20(3):363–372, 1984.

[266] Lotfi A. Zadeh. A New Direction in AI: Toward a Computational Theory of Perceptions. *AI Magazine*, 22(1): 73–84, 2001.

[267] Lotfi A. Zadeh. From Computing with Numbers to Computing with Words - From Manipulation of Measurements to Manipulation of Perceptions. *International Journal of Applied Mathematics and Computer Science (AMCS)*, 12 (3):307–324, 2002.

[268] Lotfi A. Zadeh. Precisiated natural language (pnl). *AI Magazine*, 25(3):74–91, September 2004. ISSN 0738-4602. URL `http://dl.acm.org/citation.cfm?id=1045744.1045753`.

[269] Lotfi A. Zadeh. Is there a need for fuzzy logic? *Journal of Information Sciences*, 178:2751–2779, 2008.

[270] Xiaoxiong Zhang, Bingfeng Ge, Jiang Jiang, and Yuejin Tan. Consensus building in group decision making base on multiplicative consistency with incomplete reciprocal preference relations. *Knowledge-Based Systems*, 106: 96–104, Aug 2016.

[271] Shang-Ming Zhou, Francisco Chiclana, Robert John, and Jonathan Garibaldi. Type-1 OWA operators for aggregating uncertain information with uncertain weights induced by type-2 linguistic quantifiers. *Fuzzy Sets and Systems*, 159(24):3281–3296, 2008.

[272] Shang-Ming Zhou, Francisco Chiclana, Robert John, and Jonathan Garibaldi. Alpha-level aggregation: a practical approach to type-1 OWA operation for aggregating uncertain information with applications to breast cancer treatments. *IEEE Transactions on Knowledge and Data Engineering*, 10:1455–1468, 2011.

[273] Hans-Jürgen Zimmermann. *Fuzzy set theory and its applications*. Kluwer Academic Publishers. Norwell, MA, USA, 2001.

[274] Rami Zwick, Edward Carlstein, and David V. Budescu. Measures of similarity among fuzzy concepts: A comparative analysis. *International Journal of Approximate Reasoning*, 1(2):221 – 242, 1987. ISSN 0888-613X. doi: http://dx.doi.org/10.1016/0888-613X(87)90015-6. URL `http://www.sciencedirect.com/science/article/pii/0888613X87900156`.

Part VI

Appendices

“De molen gaat niet om met wind die voorbij is. (The windmill does not care for the wind that has gone past.)”.

Dutch Anonymous

“For my part I know nothing with any certainty, but the sight of the stars makes me dream.”

Vincent van Gogh [1853-1890]

“...Serán ceniza, mas tendrá sentido; Polvo serán, mas polvo enamorado.”

Francisco de Quevedo [1580-1645]

“If I wanted you to understand it, I would have explained it better.”

Johan Cruijff [1947-2016]

“I went to one of my favourite spots under a tree. I sat there, thinking about the meaning of life. It was so warm and pleasant that I soon relaxed, dozed off, and drifted into a dream. In my dream, I found myself flying up above the field. I looked behind me and saw that I had wings. They were large and beautiful, and they fluttered rapidly. I had turned into a butterfly! It was such a feeling of freedom and joy, to be so carefree and fly around so lightly in any way I wished. Everything in this dream felt absolutely real in every way. Before long, I forgot that I was ever Chuang Tzu. I was simply the butterfly and nothing else”.

Chuang Tzu [c. 369 BC-c. 286 BC]

“When I get a little money I buy books; and if any is left I buy food and clothes”

Erasmus of Rotterdam [1466-1536]

“My paint is like a rocket, which describes its own space. I try to make the impossible possible. What is happening I cannot foresee, it is a surprise. Painting, like passion, is an emotion full of truth and rings a living sound, like the roar coming from the lion’s breast.”

Karel Appel [1921-2006]

Appendix A

Scientific contributions enabled by the student's PhD research

A.1 Journal Articles

There are a number of articles that have appeared in journals that have as a common root the research presented in this PhD Thesis. The list of articles that have appeared in Journals is the following:

Author(s)	**Publication**	**Article**
Appel, Chiclana & Carter	Acta Polytechnica Hungarica - Journal of Applied Sciences	Main Concepts, State of the Art and Future Research Questions in Sentiment Analysis [10]
Appel, Chiclana, Carter & Fujita	Knowledge-Based Systems	A Hybrid Approach to the Sentiment Analysis Problem and the Sentence Level [13]
Appel, Chiclana, Carter & Fujita	International Journal of Intelligent Systems	A Consensus Approach to the Sentiment Analysis Problem driven by Support-Based IOWA majority [17]
Appel, Chiclana, Carter & Fujita	Knowledge-Based Systems	Cross-ratio Uninorm aggregation as an enhancement to a hybrid approach to the sentiment analysis problem at the sentence level [18]
Appel, Chiclana, Carter & Fujita	Under review in Applied Intelligence (Springer); Special issue on Knowledge-Based Systems and Data Sciences	Successes and challenges in developing a hybrid approach to sentiment analysis [16]

Table A.1. Articles Published in Journals

A.2 Articles in conference's proceedings

There are a number of articles that have appeared in conference's proceedings that have as a common root the research presented in this PhD Thesis. The list of articles that have appeared in Conference Proceedings is the following:

Author(s)	**Publication**	**Article**
Appel, Chiclana, Carter & Fujita	Proceedings of the IEEE WCCI 2016 - Vancouver, Canada	A Hybrid Approach to Sentiment Analysis [12]
Appel, Chiclana, Carter & Fujita	Proceedings of IAE/AIE 2016 - Morioka, Japan	A Hybrid Approach to Sentiment Analysis with Benchmarking Results [11]
Appel, Chiclana, Carter & Fujita	Proceedings of IAE/AIE 2017 - Arras, France	A consensus approach to sentiment analysis [14]
Appel, Chiclana, Carter & Fujita	Proceedings of FUZZ-IEEE 2017 - Naples, Italy	IOWA and Cross-ratio Uninorm operators as aggregation tools in sentiment analysis and ensemble methods [15]

Table A.2. Articles Published in Conference Proceedings

Appendix B

Prototype Outputs

B.1 Main Program Output

Note: transcripts for execution of prototypes for the HSC (polarity identification) and HAC (intensity of polarity determination) models.

```
Chez Scheme (SWL) Transcript [Sun Nov 15 16:51:22 2015]
> (load "C:/Users/Orestes/Desktop/#DMUOctober2015/
     SchemeLibraryLos5000/Code/main.ss")
> (start-main)
> (transcript-off)


================================================================================
BEGINNING Results using Hybrid Method HSC.


Results for POS Dataset (Format: (POSs NEGs OBJs NOSORs ERRs):
(4367 929 0 35 0)
Results for NEG Dataset (Format: (POSs NEGs OBJs NOSORs ERRs):
(1646 3609 0 76 0)

END Results using Hybrid Method HSC.
================================================================================



================================================================================
BEGINNING Results using Hybrid Method HAC.

OUTPUT FORMAT for POSs Dataset: (POOR SLIGHTLY Moderate Very Most NOSOR Negative)
(577 1106 1006 1365 313 35 929)
OUTPUT FORMAT for NEGs Dataset: (POOR SLIGHTLY Moderate Very Most NOSOR Positive)
(770 1089 713 864 173 76 1646)


END Results using Hybrid Method HAC.
================================================================================


PERFORMANCE STATISTICS
TP = 4367
FN = 929
FP = 1646
```

```
TN = 3609

Pass One (ACC PRE REC F1):
(0.755947303573121 0.726259770497256 0.8245845921450151 0.772305243611283)

Pass Two (ACC PRE REC F1):
(0.7584880885387357 0.7278439153439153 0.825736259613581 0.7737059495562002)

================================================
Output Completed.
================================================
```

B.2 Dictionary Output

```
Chez Scheme (SWL) Transcript [Sun Nov 15 17:41:24 2015]
> (load "C:/Users/Orestes/Desktop/#DMUOctober2015/SchemeLibrary/Code/maindict.ss")
> (gendict-main)
()
> (car (cdddr (cddddr worldalfa)))
(hate 42
      (1 2 3 4 7 9 17 19 26 28 29 31 32 38 39 40 51 60 68 68 68 88
       91 91 103 109 113 118 123 123 126 133 134 135 140 140 141
       160 170 173 176 177))
> (transcript-off)


>
```

B.3 Cross-ratio Uninorm Output

Note: this is an example showing the difference values obtained when applying a Cross-Ratio Uninorm aggregation (with identity element, $e = 0.5$) against the traditional Average function.

```
Note: computing of Cross-Ratio Uninorm & Average for two elements.

================================================================================
=== Cross-ratio Uninorm Vs. Average ============================================
--------------------------------------------------------------------------------
================================================================================
================================================================================
Element x = 0.9591121579003756
Element y = 0.6124176583810436

Cross-ratio Uninorm = 0.9737288484860317

Arithmetic Mean = 0.7857649081407097

================================================================================
```

B.4 IOWA Output

Examples of application of the IOWA operator representing *consensus*. The other two methods shown are *Arithmetic mean* and *Median*.

```
================================================================================

Items to aggregate (m1 m2 m3) = (0.959112 0.500030 1.00000)
Arithmetic mean = 0.819716
Median = 0.959112
IOWA (Tolerance = 0.50) = 0.819717


Items to aggregate (m1 m2 m3) = (0.564631 0.508914 1.000000)
Arithmetic mean = 0.691181
Median = 0.564631
IOWA (Tolerance = 0.50) = 0.536773


Items to aggregate (m1 m2 m3) = (0.989550 0.682592 0.600000)
Arithmetic mean = 0.757380
Median = 0.682592
IOWA (Tolerance = 0.30) = 0.641296

================================================================================
```

Appendix C

Scheme Code - SA Hybrid System Proof of Concept

C.1 Main Program

Disclaimer: the collection of code presented in this Appendix corresponds to a prototype that was developed as a proof-of-concept for the methods being devised. The programs have not been developed for efficiency or speed, but rather as a computational mechanism to prove the validity of the proposed methods. This prototype is not fully integrated and it must be executed in parts, as needed.

```
;;
;;========= Start all calculations ====
;;=====================================
(load "C:/Users/Orestes-PC-Win10/Desktop/#DMUOctober2015/SchemeLibraryLos5000/
          Code/sorcalc.ss")
(load "C:/Users/Orestes-PC-Win10/Desktop/#DMUOctober2015/SchemeLibraryLos5000/
          Code/sorcalcv2.ss")
(load "C:/Users/Orestes-PC-Win10/Desktop/#DMUOctober2015/SchemeLibraryLos5000/
          Code/sorcalcv3.ss")
(load "C:/Users/Orestes-PC-Win10/Desktop/#DMUOctober2015/SchemeLibraryLos5000/
          Code/mecacho.ss")
;;
(define start-main
  (lambda ()
    (begin
      (calc-sor-allset-simple)
      (calc-fuzzy)
      (print-stats-pi (genlistposneg resulposz) (genlistposneg resulnegz))
      (newline)
      (display "================================================")
      (newline)
      (display "Output Completed.")
      (newline)
      (display "================================================")
      (newline)
      (newline))))
```

C.2 HSC Code

Note: some fragments of the code with perceived implementation/technique value, have been omitted as they might be commercialised in the immediate future.

```
;;
```

```
;;=================================================================
;; LOAD at once CALCULUS.ss and other utilities
(load "/Users/orestesappel/Desktop/SchemeLibrary/Code/calculus.ss")
;;=================================================================
;;=========================================
;;=== USO de las rutinas ==================
;;=========================================
;;================================
;;=== Generation of DICTIONARY ===
;;================================
;; 1. Buscar la salida OUTPUTTAGS (outtags)
;; 2. (remover-sobras (conteosi (arregla-pares (super-flatten (dicky01
;;  (saca-word-all-sin negsents) (saca-word-all-sin negsents) 1)))))
;; > ()
;; > (length worldalfa)
;; Note: worldalfa contains members of the form:
;; ((simplistic 15 (1 54 925 1182 1341 1513 ...)
;;  (silly 57 (1 142 184 275 382 508 537 ...) ...  )  ) )
;; Each member of dictionary contains:
;; (theword frequency (list of sentences where the word shows up))
;;
;;
(define corri-dict
  (lambda ()
     (remover-sobras (conteosi (arregla-pares
      (super-flatten (dicky01
      (saca-word-all-sin negsents)
      (saca-word-all-sin negsents) 1)))))))

;;
;;
;;=============================================================================
;;=== DECIDIDOR-MAX hace el primer -and simpler- SOR calculo
;; (decididor-max (filtra-initial-sor (calc-initial-sor SALIDA-DE-GETYOURBEARINGS)))
;; OR
;; (set! wow (decididor-max (filtra-initial-sor (calc-initial-sor2
;;  SALIDA-DE-GETYOURBEARINGS))))
;; (cuantos-son-pos averposs) ... cambiar "-pos" por "-neg", "-err", "-nosor"
;; & "-obj" as needed
;;=============================================================================
;;
;;=======================================
;;=== Global variables for this module ===
;;=======================================
;;
(set! resulposz '())
(set! resulnegz '())
;;
;;===========================
;;=== LOAD Dictionaries =====
;;===========================
;;
(read-file "/Users/orestesappel/Desktop/SchemeLibrary/Data/
  dicttagsnegs.ss")
```

```
(set! negdictk readbuffer)
(read-file "/Users/orestesappel/Desktop/SchemeLibrary/Data/
  dicttagsposs.ss")
(set! posdictk readbuffer)
;;=======================================
;; This variable contains all words qualified as OBJ because there
;; is no enough info
;; The format is ( (NumberOfSentence  word) ...
;; (NumberOfSentence  word) )
;;  i.e del car = (5327 nevertheless)
(set! porsiacaso '())
(set! porsiacasofull '())
(set! newwow '())
(set! newwowbest '())
(set! puchulon '())
;;
;;================================================
;;=== RUNNING the CLASSIFICATION HYBRID SYSTEM ===
;;================================================
;;
;;==============================
;;===== CALCULATE SIMPLE SOR ===
;;==============================
;;==============================
;;=== Calculate ALL Datasets ===
;;==============================
(define calc-sor-allset-simple
  (lambda ()
    (begin
      (newline)
      (display "=================================================================")
      (newline)
      (display "BEGINNING Results using Hybrid Method HSC.")
      (newline)
      (newline)
      (clear-temps)
      (newline)
      (display "Results for POS Dataset (Format: (POSs NEGs OBJs NOSORs ERRs): ")
      (newline)
      (display (calc-sor-posset-simple))
      (clear-temps)
      (newline)
      (display "Results for NEG Dataset (Format: (POSs NEGs OBJs NOSORs ERRs): ")
      (newline)
      (display (calc-sor-negset-simple))
      (newline)
      (newline)
      (display "END Results using Hybrid Method HSC.")
      (newline)
      (display "=================================================================")
      (newline)
      (newline)
      (newline))))
;;
```

```
;;

;;
;; Calculate SOR for POS Set (Output: (#pos-sor #neg-sor #obj-sor
;;  #nosor-sor #err-sor)
;; Global binding PORSIACASO, generated by "calky2"
(define calc-sor-posset-simple
  (lambda ()
    (begin
      (set! elposset (getyourbearings (limpiador-x possents)))
      (set! wow (decididor-max (filtra-initial-sor
        (calc-initial-sor2 elposset 1))))
      (set! listaobjs (extraer-obj-sennum wow 1))
      (set! listaobjsaug (dame-pal-obj-alpha
        listaobjs porsiacaso))
      (set! temp1 (sacapiojos listaobjsaug posdictk))
      (set! temp2 (consigue-negpos wow temp1))
      (set! temp3 (sacale-brillo temp2))
      (set! temp4 (pule-hebilla-parapos temp3))
      (set! temp5 (new-wow wow temp4))
      (set! resulposz newwow)
      (list (cuantos-son-pos newwow)
            (cuantos-son-neg newwow)
            (cuantos-son-obj newwow)
            (cuantos-son-nosor newwow)
            (cuantos-son-err newwow)))))

;; Calculate SOR for NEG Set (Output: (#pos-sor #neg-sor #obj-sor
;;  #nosor-sor #err-sor)
;; Global binding PORSIACASO, generated by "calky2"
(define calc-sor-negset-simple
  (lambda ()
    (begin
      (set! elnegset (getyourbearings (limpiador-x negsents)))
      (set! wow (decididor-max (filtra-initial-sor
        (calc-initial-sor2 elnegset 1))))
      (set! listaobjs (extraer-obj-sennum wow 1))
      (set! listaobjsaug (dame-pal-obj-alpha listaobjs porsiacaso))
      (set! temp1 (sacapiojos listaobjsaug negdictk))
      (set! temp2 (consigue-negpos wow temp1))
      (set! temp3 (sacale-brillo temp2))
      (set! temp4 (pule-hebilla temp3))
      (set! temp5 (new-wow wow temp4))
      (set! resulnegz newwow)
      (list (cuantos-son-pos newwow)
            (cuantos-son-neg newwow)
            (cuantos-son-obj newwow)
            (cuantos-son-nosor newwow)
            (cuantos-son-err newwow)))))

;; CLEAR Temp & Global bindings for SOR Calculations
(define clear-temps
  (lambda ()
    (set! wow '())
```

```
    (set! listaobjs '())
    (set! listaobjsaug '())
    (set! temp1 '())
    (set! temp2 '())
    (set! temp3 '())
    (set! temp4 '())
    (set! temp5 '())
    (set! temp6 '())
    (set! temp7 '())
    (set! temp8 '())
    (set! temp9 '())
    (set! temp10 '())
    (set! temp11 '())
    (set! puchulon '())
    (set! newwow '())
    (set! newwowbest '())
    (set! porsiacaso '())))


;;================================================
;;================================================
;;=== DIFERENTE APPROACH Usando PSCORE & NSCORE ===
;;================================================
;;
(define calc-sor-negset-simple-diferente
  (lambda ()
    (begin
      (set! elnegset (getyourbearings
        (limpiador-x negsents)))
      (set! wow (decididor-max (filtra-initial-sor
        (calc-initial-sor2 elnegset 1))))
      (set! listaobjs (extraer-obj-sennum wow 1))
      (set! listaobjsaug (dame-pal-obj-alpha
        listaobjs porsiacaso))
      (set! temp1 (sacapiojos listaobjsaug negdictk))
      (set! temp2 (consigue-negpos wow temp1))
      (set! temp3 (sacale-brillo temp2))
      (set! temp4 (pule-hebilla-paraposneg temp3))
      (set! temp5 (new-wow wow temp4))
      (display (list (cuantos-son-pos newwow)
            (cuantos-son-neg newwow)
            (cuantos-son-obj newwow)
            (cuantos-son-nosor newwow)
            (cuantos-son-err newwow)))
      (set! newwowbest newwow)
      (set! temp6 (jodido-obj-num newwow 1))
      (set! temp7 (jodido-obj-vecs elnegset temp6))
      (set! temp8 (letnum-vecs temp7))
      (set! temp9 (calcula-boludos temp8))
      (set! temp10 (calculadora-miobjs temp9))
      (set! temp11 (comprime-bro temp10))
      (display (list (cuantos-son-pos newwowbest)
            (cuantos-son-neg newwowbest)
            (cuantos-son-obj newwowbest)
```

```
            (cuantos-son-nosor newwowbest)
            (cuantos-son-err newwowbest)))
      )))


(define calc-sor-posset-simple-diferente
  (lambda ()
    (begin
      (set! elposset (getyourbearings
        (limpiador-x possents)))
      (set! wow (decididor-max (filtra-initial-sor
        (calc-initial-sor2 elposset 1))))
      (set! listaobjs (extraer-obj-sennum wow 1))
      (set! listaobjsaug (dame-pal-obj-alpha
        listaobjs porsiacaso))
      (set! temp1 (sacapiojos listaobjsaug posdictk))
      (set! temp2 (consigue-negpos wow temp1))
      (set! temp3 (sacale-brillo temp2))
      (set! temp4 (pule-hebilla-paraposneg temp3))
      (set! temp5 (new-wow wow temp4))
      (display (list (cuantos-son-pos newwow)
            (cuantos-son-neg newwow)
            (cuantos-son-obj newwow)
            (cuantos-son-nosor newwow)
            (cuantos-son-err newwow)))
      (set! newwowbest newwow)
      (set! temp6 (jodido-obj-num newwow 1))
      (set! temp7 (jodido-obj-vecs elposset temp6))
      (set! temp8 (letnum-vecs temp7))
      (set! temp9 (calcula-boludos temp8))
      (set! temp10 (calculadora-miobjs-pos temp9))
      (set! temp11 (comprime-bro temp10))
      (display (list (cuantos-son-pos newwowbest)
            (cuantos-son-neg newwowbest)
            (cuantos-son-obj newwowbest)
            (cuantos-son-nosor newwowbest)
            (cuantos-son-err newwowbest)))
      )))

;; takes the output of sacale-brillo and generate
;; list of the form:
;;  ( (NumSent pos/neg) (NumSent pos/neg) )
(define pule-hebilla-paraposneg
  (lambda (ls)
    (if (null? ls)
        '()
        (if (> (caar (cdr (car ls)))
          (car (cdr (car (cdr (car ls))))))
            (cons (list (caar ls) 'pos)
              (pule-hebilla-paraposneg (cdr ls)))
            (if (< (caar (cdr (car ls)))
              (car (cdr (car (cdr (car ls))))))
                (cons (list (caar ls) 'neg)
                  (pule-hebilla-paraposneg (cdr ls)))
```

```
                (cons (list (caar ls) 'obj)
                  (pule-hebilla-paraposneg (cdr ls))))))))

;; input=(neg obj pos ...); Output: (Sen1 ... SenN) for those sentences
;; labelled as OBJ (call with ct=1)
;; lat = NEWWOW
(define jodido-obj-num
  (lambda (lat ct)
    (cond
      ((null? lat) '())
      ((equal? (car lat) 'obj) (cons ct (jodido-obj-num
        (cdr lat) (+ ct 1))))
      (else (jodido-obj-num (cdr lat) (+ ct 1))))))

;; genera lista de matches with INPUT= ELNEGSET
;; SalidaDe(jodido-obj-num)
(define jodido-obj-vecs
  (lambda (neggy numby)
    (cond
      ((null? numby) '())
      (else (cons (list (car numby)
        (jodido-obj-vecs-aux neggy (car numby) 1))
                  (jodido-obj-vecs neggy
                    (cdr numby)))))))

;; auxiliar of jodido-obj-vecs
(define jodido-obj-vecs-aux
  (lambda (lista num ct)
    (cond
      ((null? lista) 'trouble)
      ((= num ct) (car lista))
      (else (jodido-obj-vecs-aux
        (cdr lista) num (+ ct 1))))))


;; toma salida de jodido-obj-vecs y elimina de la lista atomos,
;; dejando solo lista
;; de vectores con polarity labels
(define letnum-vecs
  (lambda (ls)
    (cond
      ((null? ls) '())
      (else (cons (list (caar ls)
        (letnum-vecs-aux (car (cdr (car ls)))))
        (letnum-vecs (cdr ls)))))))

;; auxiliar function to letnum-vecs
(define letnum-vecs-aux
  (lambda (lst)
    (cond
      ((null? lst) '())
      ((atom? (car lst)) (letnum-vecs-aux (cdr lst)))
      (else (cons (car lst)
        (letnum-vecs-aux (cdr lst)))))))
```

```
;; validar salida de letnum-vecs
(define seco?
  (lambda (l)
    (cond
      ((null? l) #t)
      ((and
         (list? (car l))
         (number? (caar l))
         (sopa? (car (cdr (car l)))))
      (seco? (cdr l)))
      (else #f))))

;; auxiliar de seco?
(define sopa?
  (lambda (ls)
    (cond
      ((null? ls) #t)
      ((and
         (list? (car ls))
         (vector? (caar ls))
         (atom? (car (cdr (car ls)))))
      (sopa? (cdr ls)))
      (else #f))))


;; calcular SOR de OBJs obtenidos porque Num-POSs igual Num-NEGS
;; Si no hay POS-score or NEG-score, pero
;; UPDATEINDEX en el vector es mayor de 1,
;; TRUST el Polarity label del vector
;; retorna lista de lsistas de la forma:
;; ( (numerosent ((posscore negscore updtindex PolarityLabel)
;; (posscore negscore updtindex PlarityLabel)) ... ))
(define calcula-boludos
  (lambda (ls)
    (cond
      ((null? ls) '())
      (else (cons (list (caar ls) (calcula-boludos-aux
        (car (cdr (car ls)))))
      (calcula-boludos (cdr ls)))))))

;; auxuliar de calcula-boludos
(define calcula-boludos-aux
  (lambda (l)
    (cond
      ((null? l) '())
      ((atom? (car l)) (calcula-boludos-aux (cdr ls)))
      ((vector? (caar l))
        (cons (list (getposscore (caar l)) (getnegscore (caar l))
        (getupdtindex (caar l)) (car (cdr (car l))))
        (calcula-boludos-aux (cdr l))))
      (else (calcula-boludos-aux (cdr l))))))
```

```
;; how close a number num is to number k
(define howcloseto-k
  (lambda (num bench)
    (abs (- bench num))))

;; howcloseto-0
(define howcloseto-0
  (lambda (num)
    (howcloseto-k num 0.00)))

;;howcloseto-1
(define howcloseto-1
  (lambda (num)
    (howcloseto-k num 1.00)))


;; Calcula pre-resultados basado en output de
;; "calcula-boludos"
(define calculadora-miobjs
  (lambda (ls)
    (cond
      ((null? ls) '())
      (else (cons (list (caar ls)
        (calculadora-miobjs-aux (car (cdr (car ls)))))
        (calculadora-miobjs (cdr ls)))))))

;; auxuliar to calculadora-miobjs
(define calculadora-miobjs-aux
  (lambda (l)
    (cond
      ((null? l) '())
      ((= (length l) 1)
        (if (and
          (number? (caar l))
          (number? (car (cdr (car l)))))
        (if (> (caar l) (car (cdr (car l))))
          'pos
          'neg)
        (car (cdddr (car l)))))
      (else (contalos-todos-pana
        (calculadora-miobjs-aux-aux l) 0 0)))))


;; retorna lista de (neg pos neg pos) luego de aplicar
;; los criterios apropiados
(define calculadora-miobjs-aux-aux
  (lambda (l)
    (cond
      ((null? l) '())
      ((and
         (number? (caar l))
         (number? (car (cdr (car l)))))
      (if (> (caar l) (car (cdr (car l))))
        (cons 'pos
```

```
          (calculadora-miobjs-aux-aux (cdr l)))
        (cons 'neg
          (calculadora-miobjs-aux-aux (cdr l)))))
      (else (cons (car (cdddr (car l)))
        (calculadora-miobjs-aux-aux (cdr l)))))))


;; Calcula pre-resultados basado en output de "calcula-boludos"
;; (para POS)
(define calculadora-miobjs-pos
  (lambda (ls)
    (cond
      ((null? ls) '())
      (else (cons (list (caar ls)
        (calculadora-miobjs-aux-pos (car (cdr (car ls)))))
        (calculadora-miobjs-pos (cdr ls)))))))

;; auxuliar to calculadora-miobjs (para POS)
(define calculadora-miobjs-aux-pos
  (lambda (l)
    (cond
      ((null? l) '())
      ((= (length l) 1)
        (if (and
          (number? (caar l))
          (number? (car (cdr (car l)))))
        (if (>= (caar l) (car (cdr (car l))))
          'pos
          'neg)
        (car (cdddr (car l)))))
      (else (contalos-todos-pana
        (calculadora-miobjs-aux-aux-pos l) 0 0)))))


;; retorna lista de (neg pos neg pos) luego de aplicar los criterios
;; apropiados (para POS)
(define calculadora-miobjs-aux-aux-pos
  (lambda (l)
    (cond
      ((null? l) '())
      ((and
         (number? (caar l))
         (number? (car (cdr (car l)))))
      (if (>= (caar l) (car (cdr (car l))))
        (cons 'pos
          (calculadora-miobjs-aux-aux-pos (cdr l)))
        (cons 'neg
          (calculadora-miobjs-aux-aux-pos (cdr l)))))
      (else (cons (car (cdddr (car l)))
        (calculadora-miobjs-aux-aux-pos (cdr l)))))))

;;-------------------------------------------------
```

```
;; contalos-todos-pana (cuenta los NEGs and POSs. Llamar con
;;  (l pneggy cneggy), cneggy & pneggy = 0
(define contalos-todos-pana
  (lambda (l cneggy cpoggy)
    (cond
      ((null? l) (if (> cpoggy cneggy) 'pos 'neg))
      ((equal? (car l) 'pos) (contalos-todos-pana
        (cdr l) cneggy (+ cpoggy 1)))
      (else (contalos-todos-pana (cdr l)
        (+ cneggy 1) cpoggy)))))

;; comprime-bro: reemplaza NEGs/POSs as per list "ls" y resulatdo
;; de NEWWOW; ct=1 es el contador
(define comprime-bro
  (lambda (ls)
    (cond
      ((null? ls) '())
      (else (begin
              (set! newwowbest
                (comprime-bro-aux (car ls) newwowbest 1))
              (comprime-bro (cdr ls)))))))

;; auxiliar to comprime-bro
(define comprime-bro-aux
  (lambda (parinf bdd ct)
    (cond
      ((null? bdd) '())
      ((= (car parinf) ct) (cons (car (cdr parinf))
        (comprime-bro-aux parinf
        (cdr bdd) (+ ct 1))))
      (else (cons (car bdd)
        (comprime-bro-aux parinf (cdr bdd)
        (+ ct 1)))))))




;;=====================================================================
;;=======================================
;;
;;=======================================
;;=====================================================================
;;==== MAIN CALL: Initial Case of CALCULATING SOR ======================
;; (decididor-max (filtra-initial-sor (calc-initial-sor
;;  SALIDA-DE-GETYOURBEARINGS 1)))
;; (cuantos-son-pos averposs) ... cambiar -pos por -neg, -err,
;; -nosor & -obj
;;=====================================================================
;;
;;=========================================================
;;=== Module for estimation pos/neg of set of sentences ===
;;=========================================================
(define calc-initial-sor
  (lambda (ls)
```

```
    (cond
      ((null? ls) '())
      ((lista-atomica? (car ls))
        (cons 'nosor (calc-initial-sor (cdr ls))))
      (else (cons (calky (car ls))
        (calc-initial-sor (cdr ls)))))))

;; calc-initial-sor2 Version 2: call like (calc-initial-sor2 ls 1), ctd = 1
(define calc-initial-sor2
  (lambda (ls ctd)
    (cond
      ((null? ls) '())
      ((lista-atomica? (car ls))
        (cons 'nosor
          (calc-initial-sor2 (cdr ls) (+ ctd 1))))
      (else
        (cons (calky2 (car ls) ctd)
          (calc-initial-sor2 (cdr ls)
            (+ ctd 1)))))))


;; auxiliar of calky
(define goodyone-4calc?
  (lambda (ls)
    (and
      (list? ls)
      (vector? (car ls))
      (or
        (equal? (car (cdr ls)) 'pos)
        (equal? (car (cdr ls)) 'neg)
        (equal? (car (cdr ls)) 'obj)))))


;; retorna lista de either NOSORs or (cp cn co)
(define filtra-initial-sor
  (lambda (l)
    (cond
      ((null? l) '())
      ((equal? (car l) 'nosor)
        (cons 'nosor (filtra-initial-sor (cdr l))))
      ((list? (car l)) (cons (calculador-y (car l) 0 0 0)
        (filtra-initial-sor (cdr l))))
      (else (cons 'error (filtra-initial-sor (cdr l)))))))

;;auxiliar de filtra-initial-sor
(define calculador-y
  (lambda (l cp cn co)
    (cond
      ((null? l) (list cp cn co))
      ((equal? (car l) 'obj)
        (calculador-y (cdr l) cp cn (+ co 1)))
      ((equal? (car l) 'neg)
        (calculador-y (cdr l) cp (+ cn 1) co))
      ((equal? (car l) 'pos)
```

```
        (calculador-y (cdr l) (+ cp 1) cn co))
      (else (calculador-y (cdr l) cp cn co)))))

;; macro decididor
(define decididor-max
  (lambda (ls)
    (cond
      ((null? ls) '())
      ((equal? (car ls) 'nosor)
        (cons 'nosor (decididor-max (cdr ls))))
      (else (cons (decididor (car ls))
        (decididor-max (cdr ls)))))))


;; decide el SOR de la oracion generando either POS or NEG:
;; auxiliar de decididor-max
(define decididor
  (lambda (l)
    (cond
      ((null? l) 'error)
      ((> (car l) (car (cdr l))) 'pos)
      ((< (car l) (car (cdr l))) 'neg)
;;      ((= (car l) (car (cdr l))) 'obj)
        ((= (car l) (car (cdr l))) 'obj)
      (else 'undecided))))

;; cuenta cuantos fueron POS en resultado de DECIDIDOR-MAX
(define cuantos-son-pos
  (lambda (l)
    (if (null? l)
        0
        (if (equal? (car l) 'pos)
            (+ 1 (cuantos-son-pos (cdr l)))
            (cuantos-son-pos (cdr l))))))

;; cuenta cuantos fueron NEG en resultado de DECIDIDOR-MAX
(define cuantos-son-neg
  (lambda (l)
    (if (null? l)
        0
        (if (equal? (car l) 'neg)
            (+ 1 (cuantos-son-neg (cdr l)))
            (cuantos-son-neg (cdr l))))))

;; cuenta cuantos fueron OBJ en resultado de DECIDIDOR-MAX
(define cuantos-son-obj
  (lambda (l)
    (if (null? l)
        0
        (if (equal? (car l) 'obj)
            (+ 1 (cuantos-son-obj (cdr l)))
            (cuantos-son-obj (cdr l))))))

;; Genera lista de los numeros de las oraciones que resultaron OBJ.
```

```
;; Numor=numero de oracion
(define cuales-son-obj
  (lambda (l numor)
    (if (null? l)
        '()
        (if (equal? (car l) 'obj)
            (cons numor (cuales-son-obj (cdr l) (+ numor 1)))
            (cuales-son-obj (cdr l) (+ numor 1))))))


;; cuenta cuantos fueron ERROR en resultado de DECIDIDOR-MAX
(define cuantos-son-err
  (lambda (l)
    (if (null? l)
        0
        (if (equal? (car l) 'error)
            (+ 1 (cuantos-son-err (cdr l)))
            (cuantos-son-err (cdr l))))))

; cuenta cuantos fueron NOSOR en resultado de DECIDIDOR-MAX
(define cuantos-son-nosor
  (lambda (l)
    (if (null? l)
        0
        (if (equal? (car l) 'nosor)
            (+ 1 (cuantos-son-nosor (cdr l)))
            (cuantos-son-nosor (cdr l))))))

;;=================================================================
;;=== Instrucciones para conseguir resultados para
;;      las oraciones OBJ ===
;;=================================================================
;;
;;
;; generate list with number of sentences of works classified as OBJ
(define extraer-obj-sennum
  (lambda (ls ct)
    (if (null? ls)
        '()
        (if (equal? (car ls) 'obj)
            (cons ct (extraer-obj-sennum (cdr ls) (+ ct 1)))
            (extraer-obj-sennum (cdr ls) (+ ct 1))))))

;; returns list of form ( (Num1 (w1 w2 ... wk)) ...
;;  (NumK (wk wj ... wn)) ):
;; where Wk are works classified as OBJ.
(define dame-pal-obj-alpha
  (lambda (l ls)
    (cond
      ((null? l) '())
      (else (cons (list (car l)
        (dame-pal-obj-alpha-prima (car l) ls))
        (dame-pal-obj-alpha (cdr l) ls))))))
```

```
;; auxiliar of dame-pal-obj-alpha
(define dame-pal-obj-alpha-prima
  (lambda (item ls)
    (cond
      ((null? ls) '())
      ((= item (caar ls)) (cons (car (cdr (car ls)))
        (dame-pal-obj-alpha-prima item (cdr ls))))
      (else (dame-pal-obj-alpha-prima item (cdr ls))))))

;; returns list of the form
;; ( (numberSent (sentNum ... sentNumb )) ... )
(define sacapiojos
  (lambda (laug dictk)
    (cond
      ((null? laug) '())
      (else (cons (list (caar laug)
        (super-flatten (sacapiojos-aux
        (car (cdr (car laug))) dictk)))
      (sacapiojos (cdr laug) dictk))))))

;; auxiliar to scapiojos
(define sacapiojos-aux
  (lambda (l dicc)
    (cond
      ((null? l) '())
      (else (cons (sacapiojos-aux-aux (car l) dicc)
        (sacapiojos-aux (cdr l) dicc))))))

;; auxiliar of sacapiojos-aux
(define sacapiojos-aux-aux
  (lambda (item dicc)
    (if (equal? item (caar dicc))
        (car (cddr (car dicc)))
        (sacapiojos-aux-aux item (cdr dicc)))))

;; generate list of the form:
;; ( (NumOfSent (obj neg obj pos neg pos pos neg neg neg neg)) ... )
(define consigue-negpos
  (lambda (lat ls)
    (cond
      ((null? ls) '())
      (else (cons (list (caar ls)
        (super-flatten (consigue-negpos-aux
        (car (cdr (car ls))) lat)))
      (consigue-negpos lat (cdr ls)))))))

;; auxiliar to consigue-nespos
(define consigue-negpos-aux
  (lambda (l lat)
    (cond
      ((null? l) '())
      (else (cons (consigue-negpos-aux2 (car l) lat l)
        (consigue-negpos-aux (cdr l) lat))))))
```

```
;; auxiliar to consigue-negpos-aux
(define consigue-negpos-aux2
  (lambda (numb lat ct)
    (cond
      ((null? lat) '())
      ((= numb ct) (cons (car lat)
        (consigue-negpos-aux2 numb (cdr lat) (+ ct 1))))
      (else (consigue-negpos-aux2 numb (cdr lat) (+ ct 1))))))

;; returns a list of the form: ( (numSent (countpos countneg)) ... )
(define sacale-brillo
  (lambda (ls)
    (cond
      ((null? ls) '())
      (else (cons (list (caar ls) (list (cuenta-pos-uhu
        (car (cdr (car ls))))
      (cuenta-neg-uhu (car (cdr (car ls))))))
      (sacale-brillo (cdr ls)))))))

;; count the POS occurrences
(define cuenta-pos-uhu
  (lambda (l)
    (if (null? l)
        0
        (if (equal? (car l) 'pos)
            (+ 1 (cuenta-pos-uhu (cdr l)))
            (cuenta-pos-uhu (cdr l))))))

;; count the NEG occurrences
(define cuenta-neg-uhu
  (lambda (l)
    (if (null? l)
        0
        (if (equal? (car l) 'neg)
            (+ 1 (cuenta-neg-uhu (cdr l)))
            (cuenta-neg-uhu (cdr l))))))

;; takes the output of sacale-brillo and
;; generate list of the form:
;; ( (NumSent pos/neg) (NumSent pos/neg) )
;;  OJOOJOOJOOJOOJOOJOOJO
(define pule-hebilla
  (lambda (ls)
    (if (null? ls)
        '()
        (if (> (caar (cdr (car ls)))
          (car (cdr (car (cdr (car ls))))))
            (cons (list (caar ls) 'pos)
              (pule-hebilla (cdr ls)))
            (cons (list (caar ls) 'neg)
              (pule-hebilla (cdr ls)))))))

;; takes the output of sacale-brillo and generate list of the form:
;; ( (NumSent pos/neg) (NumSent pos/neg) )
```

```
(define pule-hebilla-parapos
  (lambda (ls)
    (if (null? ls)
        '()
        (if (>= (caar (cdr (car ls)))
          (car (cdr (car (cdr (car ls))))))
            (cons (list (caar ls) 'pos)
              (pule-hebilla-parapos (cdr ls)))
            (cons (list (caar ls) 'neg)
              (pule-hebilla-parapos (cdr ls)))))))



;; takes salida de pule-hebilla and generates new list de resultados
;; with only pos/neg particles
(define new-wow
  (lambda (oldwow newcand)
    (cond
      ((null? newcand) '())
      (else (begin
              (set! newwow (new-wow-aux oldwow (car newcand) 1))
              (new-wow newwow (cdr newcand)))))))

;;
(define new-wow-aux
  (lambda (oldl l ct)
    (cond
      ((null? oldl) '())
      ((equal? (car l) ct) (cons (car (cdr l))
        (new-wow-aux (cdr oldl) l (+ ct 1))))
      (else (cons (car oldl)
        (new-wow-aux (cdr oldl) l (+ ct 1)))))))


;;============================
;;============ END ==========
;;============================

;;
(define group-porsiacaso
  (lambda (ls)
    (if (null? ls)
        '()
        (cons (list (car (cdr (car ls)))
          (group-porsi-aux (car (cdr (car ls))) ls))
          (group-porsiacaso (cdr ls))))))

;; auxiliar of group-porsiacaso
(define group-porsi-aux
  (lambda (item ls)
    (cond
      ((null? ls) '())
      ((equal? item (car (cdr (car ls))))
        (cons (caar ls)
```

```
        (group-porsi-aux item (cdr ls))))
      (else (group-porsi-aux item (cdr ls))))))

;; ver si hay palabras repretidas en la salida
;; de "group-porsiacaso"
(define bastards-rep?
  (lambda (ls)
    (cond
      ((null? ls) #f)
      ((member-weird-case? (caar ls)
        (cdr ls)) #t)
      (else (bastards-rep? (cdr ls))))))

(define bastards-rep-print
  (lambda (ls)
    (cond
      ((null? ls) '())
      ((member-weird-case?
        (caar ls) (cdr ls)) (cons
        (member-weird-case-print
          (caar ls) (cdr ls))
        (bastards-rep-print (cdr ls))))
      (else (bastards-rep-print (cdr ls))))))

(define member-weird-case?
  (lambda (item ls)
    (cond
      ((null? ls) #f)
      ((equal? item (caar ls)) #t)
      (else
        (member-weird-case? item (cdr ls))))))
;;
(define member-weird-case-print
  (lambda (item ls)
    (cond
      ((null? ls) '())
      ((equal? item (caar ls))
        (cons (car ls)
        (member-weird-case-print item (cdr ls))))
      (else (member-weird-case-print item (cdr ls))))))



;;====================================
;; calculate performance indicators
;; INPUT: TP, FP, FN, TN
;; OUTPUT: (A P R F1)
;;====================================
(define indc-confusion
  (lambda (tp fp fn tn)
    (append (indc-confusion1 tp fp fn tn)
      (indc-confusion2
      (indc-confusion1 tp fp fn tn)))))
```

```
(define indc-confusion1
  (lambda (tp fp fn tn)
    (list (/ (+ tp tn) (+ (+ tp (+ tn fp)) fn))
          (/ tp (+ tp fp))
          (/ tp (+ tp fn)))))

(define indc-confusion2
  (lambda (l)
    (list (/ (* 2 (* (car (cdr l)) (car (cddr l))))
      (+  (car (cdr l))  (car (cddr l)))))))

;;=====================
;;=== Version TWO =====
;;=====================
(define indc2-confusion2
  (lambda (tp fp fn tn)
    (append (indc2-confusion21 tp fp fn tn)
      (indc2-confusion22
        (indc2-confusion21 tp fp fn tn)))))

(define indc2-confusion21
  (lambda (tp fp fn tn)
    (list (/ (+ tp (+ tn 111))
      (+ (+ tp (+ tn fp)) (+ fn 111)))
          (/ (+ tp 35) (+ tp (+ fp 35)))
          (/ (+ tp 35) (+ tp (+ 35 fn))))))

(define indc2-confusion22
  (lambda (l)
    (list (/ (* 2 (* (car (cdr l)) (car (cddr l))))
      (+  (car (cdr l))  (car (cddr l)))))))

;;
;;============================
;;=== Manejar OBJs ===========
;;============================
;; Llamar como (dame-losobj7 negsset/posset low 1)
(define dame-losobj7
  (lambda (ls low ct)
    (cond
      ((null? low) '())
      ((= (car low) ct) (cons (dame-losob7-aux ls
        (car low))
      (dame-losobj7 ls (cdr low) (+ ct 1))))
      (else (dame-losobj7 ls (cdr low) 1)))))

;;=========================
;;=== END Manejar OBJs ===
;;=========================
;;
;;
;;================================================
;;=== START statistics for scores of Lexicons ===
;;================================================
```

```
;;
;; returns the number of posscore that are numbers
;;  AND different from zero
(define cuantos-sonnum-poslex
  (lambda (ls)
    (cond
      ((null? ls) 0)
      ((and
         (number? (getposscore (caar ls)))
         (not (= (getnegscore (caar ls)) 0.0)))
       (+ 1 (cuantos-sonnum-poslex (cdr ls))))
      (else (cuantos-sonnum-poslex (cdr ls))))))


;; returns the number of negscore that are numbers AND
;; different from zero for a given lexicon "ls"
(define cuantos-sonnum-neglex
  (lambda (ls)
    (cond
      ((null? ls) 0)
      ((and
         (number? (getnegscore (caar ls)))
         (not (= (getnegscore (caar ls)) 0.0)))
       (+ 1 (cuantos-sonnum-neglex (cdr ls))))
      (else (cuantos-sonnum-neglex (cdr ls))))))

;; returns the number of negscore that are numbers AND
;; different from zero for a given lexicon "ls"
(define cuantos-sonnum-objlex
  (lambda (ls)
    (cond
      ((null? ls) 0)
      ((and
         (number? (getobjscore (caar ls)))
         (not (= (getobjscore (caar ls)) 0.0)))
       (+ 1 (cuantos-sonnum-objlex (cdr ls))))
      (else (cuantos-sonnum-objlex (cdr ls))))))

;; returns the summation of posscores that are numbers AND
;; different from zero for a given lexicon "ls"
(define sumar-posscore
  (lambda (ls)
    (cond
      ((null? ls) 0)
      ((number? (getposscore (caar ls)))
        (+ (getposscore (caar ls))
          (sumar-posscore (cdr ls))))
      (else (sumar-posscore (cdr ls))))))

;; returns the summation of negscores that are numbers AND
;; different from zero for a given lexicon "ls"
(define sumar-negscore
  (lambda (ls)
    (cond
```

```
      ((null? ls) 0)
      ((number? (getnegscore (caar ls)))
        (+ (getnegscore (caar ls))
          (sumar-negscore (cdr ls))))
      (else (sumar-negscore (cdr ls))))))

;; returns the summation of objscores that are numbers
;; for a given lexicon "ls"
(define sumar-objscore
  (lambda (ls)
    (cond
      ((null? ls) 0)
      ((number? (getobjscore (caar ls)))
        (+ (getobjscore (caar ls))
          (sumar-objscore (cdr ls))))
      (else (sumar-objscore (cdr ls))))))

;; return max POSSCORE of a given lexicon "ls"
;; called first time with max = -1.0,
;; funk = {getposscore, getnegscore, getobjscore}
(define dame-max-lex
  (lambda (ls max)
    (if (null? ls)
        max
        (if (number? (getobjscore (caar ls)))
            (if (> (getobjscore (caar ls)) max)
                (dame-max-lex (cdr ls)
                  (getobjscore (caar ls)))
                (dame-max-lex (cdr ls) max))
            (dame-max-lex (cdr ls) max)))))

;; return min POSSCORE of a given lexicon "ls"
;; call first time win min = 99.0,
;; funk = {getposscore, getnegscore, getobjscore}
(define dame-min-lex
  (lambda (ls min)
    (if (null? ls)
        min
        (if (number? (getobjscore (caar ls)))
            (if (< (getobjscore (caar ls)) min)
                (dame-min-lex (cdr ls)
                  (getobjscore (caar ls)))
                (dame-min-lex (cdr ls) min))
            (dame-min-lex (cdr ls) min)))))

;;=========================================
;;=== END statistics scores of LEXICONS ===
;;=========================================
;;

;;=========================================
;;=== Rebuild PORSIACASO ==================
;;=== (porsiacaso porsiacaso porsiacaso 1) ===
;;=========================================
```

```
(define buildporsi
  (lambda (ct ls)
    (cond
      ((= ct 183) '())
      (else
        (cons (list ct (buildporsi-aux ct ls))
        (buildporsi (+ ct 1) ls))))))

;; auxiliar for BUILDPORSI
(define buildporsi-aux
  (lambda (c l)
    (cond
      ((null? l) '())
      ((= (caar l) c) (cons (car (cdr (car l)))
        (buildporsi-aux c (cdr l))))
      (else (buildporsi-aux c (cdr l))))))

;; get a list with the sentence numbers that were not OBJ
(define tellblankyss
  (lambda (ls)
    (cond
      ((null? ls) '())
      ((equal? (car (cdr (car ls))) '())
        (cons (caar ls) (tellblankyss (cdr ls))))
      (else (tellblankyss (cdr ls))))))


;;================================== AUXILIAR PROCEDURES =============
;;
;;==========================
;;=== SNOC (mysnoc) ========
;;==========================
(define mysnoc
  (lambda (item lst)
    (cond
      ((null? lst) (cons item '()))
      (else (cons (car lst) (mysnoc item (cdr lst)))))))

;;================================== END AUXILIAR PROCS ==============
;;
;;=====================================================================
;;=== Build a dictionario or WORD, #times shows in set, and in which
;; sentencces the word shows up ===
;;=====================================================================
;; Creo que: Receives SalidaDeBuildporsi, lista de poss/negs (5331),
;; negsents & possents (aplanados)
;;
;;=====================================================================
;; aplicar a la salida de 'saca-word-all-sin'
;;=====================================================================
(define dicky01
  (lambda (ls lals num)
    (cond
      ((null? ls) '())
```

```
      (else (cons (dicky01-aux (car ls) lals num)
        (dicky01 (cdr ls) lals (+ num 1)))))))

;; auxiliar to dicky01
;; devuelve lista de la forma:
;; "( (W1 frequencyOfW1 ListOfSentenceNumbersWhereW1Shows) .... )"
(define dicky01-aux
  (lambda (l ls numsent)
    (cond
      ((null? l) '())
      ((member (car l) ls) (cons (list (car l) (member-lista-all
        (car l) ls numsent)) (dicky01-aux (cdr l) ls numsent)))
      (else (cons (list (car l) (list numsent))
        (dicky01-aux (cdr l) ls numsent))))))

;; member-all?
(define member-all?
  (lambda (item ls)
    (if (null? ls)
        #f
        (or
          (equal? (car ls) item)
          (and
            (not (pair? (car ls)))
            (member-all? item (cdr ls)))
          (and
            (pair? (car ls))
            (or
              (member-all? item (car ls))
              (member-all? item (cdr ls))))))))


;; member-counter-all
(define member-counter-all
  (lambda (item ls)
    (cond
      ((null? ls) 1)
      ((equal? (car ls) item)
        (+ 1 (member-counter-all item (cdr ls))))
      (else (member-counter-all item (cdr ls))))))

;; member-lista-all
(define member-lista-all
  (lambda (item ls num)
    (cond
      ((null? ls) '())
      ((equal? (car ls) item) (cons num
        (member-lista-all item (cdr ls) (+ num 1))))
      (else (member-lista-all item (cdr ls) num)))))

;;
(define weird-flatten
  (lambda (lst)
    (cond
```

```
      ((null? lst) '())
      ((atom? (caar lst)) (cons (car lst)
        (weird-flatten (cdr lst))))
      (else (cons (weird-flatten (car lst))
                  (weird-flatten (cdr lst)))))))


(define weird-atom?
  (lambda (x)
    (and (not (pair? x)) (not (null? x)))))


;; construye pares "( (palabra OracionDondeAparece) ... )"
(define arregla-pares
  (lambda (ls)
    (if (odd? (length ls))
        '()
        (arregla-pares-aux ls))))

;; auxuliar de arregla-pares
(define arregla-pares-aux
  (lambda (ls)
    (cond
      ((null? ls) '())
      (else (cons (list (car ls) (car (cdr ls)))
        (arregla-pares-aux (cddr ls)))))))


;; member para lista de pares de la forma
;; "( (w1 number) (w2 number) ... (wN number) )", pero cuenta
(define member-pares-cnt
  (lambda (item lsp)
    (cond
      ((null? lsp) 0)
      ((equal? (caar lsp) item)
       (+ 1 (member-pares-cnt item (cdr lsp))))
      (else (member-pares-cnt item (cdr lsp))))))

;;
(define member-pares-mklst
  (lambda (item lsp)
    (cond
      ((null? lsp) '())
      ((equal? (caar lsp) item) (cons (car (cdr (car lsp)))
        (member-pares-mklst item (cdr lsp))))
      (else (member-pares-mklst item (cdr lsp))))))


;; genera vector (palabra freq (n1 n2 n3)) con repeticiones

(define conteosi
  (lambda (ls)
    (cond
      ((null? ls) '())
```

```
      (else (cons (list (caar ls)
        (member-pares-cnt (caar ls) ls)
        (member-pares-mklst (caar ls) ls))
      (conteosi (cdr ls)))))))

;; VARIABLE RESULTADO sin repeticion del DICCIONARIO
(set! worldalfa '())

;; remover ocurrencias repetidas
(define remover-sobras
  (lambda (ls)
    (cond
      ((null? ls) '())
      (else
        (begin
          (set! worldalfa (snoc (car ls) worldalfa))
          (remover-sobras (des-hacer (car ls)
            (cdr ls))))))))

;; auxiliar de remover-sobras
(define des-hacer
  (lambda (litem l)
    (cond
      ((null? l) '())
      ((equal? (caar l) (car litem))
        (des-hacer litem (cdr l)))
      (else (cons (car l)
        (des-hacer litem (cdr l)))))))



;;
;;============================
;;=== flatten (super) ========
;;============================
(define super-flatten
  (lambda (ls)
    (cond
      ((null? ls) '())
      ((pair? (car ls))
        (append (super-flatten (car ls))
          (super-flatten (cdr ls))))
      (else (cons (car ls)
        (super-flatten (cdr ls)))))))

;;
;;=============================================================
;;=== obtain flat list of words from Output of Parser =======
;;=== returning ( (NS1 (w1 w2 wn) ... (NSn (w1 w2 wn)) ) ====
;;=============================================================
(define saca-word-all
  (lambda (ls num)
    (cond
      ((null? ls) '())
```

```
      (else (cons (list num (despojar-z (car ls)))
        (saca-word-all (cdr ls) (+ num 1)))))))

;; auxiliar of saca-word-all
(define despojar-z
  (lambda (ls)
    (cond
      ((null? ls) '())
      (else (cons (caar ls) (despojar-z (cdr ls)))))))

;; same as saca-word-all pero sin NSn
(define saca-word-all-sin
  (lambda (ls)
    (cond
      ((null? ls) '())
      (else (cons (despojar-z (car ls))
        (saca-word-all-sin (cdr ls)))))))

;;-----------------------------------------
;; Load Libraries Section ----------------
;;-----------------------------------------
(load "/Users/orestesappel/Desktop/SchemeLibrary/Code/miiodec2014.ss")
(load "/Users/orestesappel/Desktop/SchemeLibrary/Code/various.ss")
(load "/Users/orestesappel/Desktop/SchemeLibrary/Code/prefilter.ss")
(load "/Users/orestesappel/Desktop/SchemeLibrary/Code/lexeditor.ss")
(load "/Users/orestesappel/Desktop/SchemeLibrary/Code/iowacode23nov2014.ss")
;;-----------------------------------------
;;
;-----------------------------------------
;; Initialise Global Bindings ------------
;;-----------------------------------------
(read-file "/Users/orestesappel/Desktop/SchemeLibrary/Data/
␣␣mylexposnegobjv7.ss")
(set! thelex readbuffer)
;;
(read-file "/Users/orestesappel/Desktop/SchemeLibrary/Data/
␣␣mylexnegsv7.ss")
(set! neglex readbuffer)
;;
(read-file "/Users/orestesappel/Desktop/SchemeLibrary/Data/
␣␣mylexpossv7.ss")
(set! poslex readbuffer)
;;
(read-file "/Users/orestesappel/Desktop/SchemeLibrary/Data/
␣␣mylexobjsv7.ss")
(set! objlex readbuffer)
;;
(read-file "/Users/orestesappel/Desktop/SchemeLibrary/Data/
␣␣outputtagsposv7.txt")
(set! possents readbuffer)
;;
(read-file "/Users/orestesappel/Desktop/SchemeLibrary/Data/
␣␣outputtagsnegv7.txt")
(set! negsents readbuffer)
;;
```

```
;;
(read-file "/Users/orestesappel/Desktop/SchemeLibrary/Data/sentencedata/
  sentencespos.txt")
(set! posorigs readbuffer)
;;
(read-file "/Users/orestesappel/Desktop/SchemeLibrary/Data/sentencedata/
  sentencesneg.txt")
(set! negorigs readbuffer)
;;
;;(read-file "/Users/orestesappel/Desktop/SchemeLibrary/Data/sentencedata/
              sentencesobj.txt")
;;(set! objorigs readbuffer)
;;-----------------------------------------
;;
;;-------------------------------------------------------------------------
;; Programs for Calculating Semantic Orientation of a List of Sentences ---
;;-------------------------------------------------------------------------

;; Returns a list with only objects from lexicon or 'nowordsforsentence' symbol
(define getvecsonly
  (lambda (lst)
    (if (null? lst)
        '()
        (if (null? (getvecsonly-aux (car lst)))
            (cons 'nowordsforsentence (getvecsonly (cdr lst)))
            (cons (getvecsonly-aux (car lst))
                  (getvecsonly (cdr lst)))))))

;; Aux function to getvecsonly (works at the sentence level)
(define getvecsonly-aux
  (lambda (ls)
    (cond
      ((null? ls) '())
      ((symbol? (car ls)) (getvecsonly-aux (cdr ls)))
      ((and
        (list? (car ls))
        (vector? (caar ls))) (cons (car ls)
        (getvecsonly-aux (cdr ls))))
      (else (getvecsonly-aux (cdr ls))))))

;;-------------------------------------------------------------------------
;; Some GLOBAL support functions ----------------------------------------
;;-------------------------------------------------------------------------
;; yields the number of sentences coming out of 'getyourbearings'
;; with "notermsnvar"
(define versy
  (lambda (l)
    (cond
      ((null? l) 0)
      ((equal? (car l) 'notermsnvar) (+ (versy (cdr l)) 1))
      (else (versy (cdr l))))))

;; like versy, but returns a list with all sentence numbers instead
(define versy2
```

```
    (lambda (l c)
      (cond
        ((null? l) '())
        ((equal? (car l) 'notermsnvar) (cons c (versy2 (cdr l)
          (+ c 1))))
        (else (versy2 (cdr l) (+ c 1)))))


;; obtain original sentence given the sentence number
;; sentence numbers come in a list of integers; type = 'n' or 'p'
(define sacasenty
  (lambda (ls type)
    (cond
      ((null? ls) '())
      (else (if (equal? type 'n)
                (cons (sacasenty-aux (car ls) negorigs 1)
                      (sacasenty (cdr ls) type))
                (cons (sacasenty-aux (car ls) posorigs 1)
                      (sacasenty (cdr ls) type)))))))

;; auxuliar function de sacasenty
(define sacasenty-aux
  (lambda (numm losorig cursor)
    (cond
      ((null? losorig) '())
      ((= numm cursor) (cons (car losorig)
        (sacasenty-aux numm
          (cdr losorig) (+ cursor 1))))
      (else (sacasenty-aux numm (cdr losorig) (+ cursor 1))))))

;; return oracion X de la salida de "geytourbearings"
(define oracion-x
  (lambda (lst target count)
    (cond
      ((null? lst) 'out-of-range)
      ((= target count) (car lst))
      (else (oracion-x (cdr lst) target (+ count 1))))))

;; List all PoS tags in the list of sentenes generated in Python
(define getmetagsposp
  (lambda (lst)
    (cond
      ((null? lst) '())
      (else (cons (getmetagsposp-aux (car lst)) (getmetagsposp
        (cdr lst)))))))

;; aux for getmetagsposp
(define getmetagsposp-aux
  (lambda (l)
    (cond
      ((null? l) '())
      (else (cons (car (cdr (car l))) (getmetagsposp-aux
        (cdr l)))))))
```

```
;; flatten a given list
(define myflatten-x
    (lambda (ls)
      (cond
        ((null? ls) '())
        ((pair? (car ls)) (append (myflatten-x (car ls))
                                  (myflatten-x (cdr ls))))
        (else (cons (car ls) (myflatten-x (cdr ls)))))))

;; remove duplicates from a flat list of symbols
(define remdupsmyflatten
  (lambda (ls)
    (cond
      ((null? ls) '())
      ((> (length ls) 1) (if (member (car ls) (cdr ls))
                             (remdupsmyflatten (cdr ls))
                             (cons (car ls)
                                   (remdupsmyflatten
                                    (cdr ls)))))
      (else (cons (car ls) (remdupsmyflatten
                            (cdr ls)))))))

;; dsds
(define wholething-x
  (lambda (ls)
    (remdupsmyflatten (myflatten-x (getmetagsposp ls)))))



;; chage the pos score and negscore for list of
;; the form (v1 neg)
(define cambiascores-neglex
  (lambda (ls)
    (cond
      ((null? ls) '())
      ((equal? (car (cdr (car ls))) 'neg)
       (cons (cambiascores-aux
              (car ls)) (cambiascores-neglex (cdr ls))))
      (else (cons (car ls) (cambiascores-neglex
                            (cdr ls)))))))

;; auxiliar function of cambiascores-neglex
(define cambiascores-aux
  (lambda (l)
    (list (list->vector (list (getword (car l))
                              (getpos (car l))
                              (getnegscore (car l))
                              (getposscore (car l))
                              (getobjscore (car l))
                              (getsor (car l))
                              (getmaxdist (car l))
                              (getmindist (car l))
                              (getupdtindex
                               (car l))))))
```

```
                                (car (cdr l)))))

;;==========================================
;;=== check contents after getyourbearings ===
;;==========================================
;; devuleve lista de oraciones que no tienen ni un solo vector,
;; con el numero de oracion
(define afterbearings
  (lambda (ls ct)
    (cond
      ((null? ls) '())
      ((and
        (list? (car ls))
        (todosatoms? (car ls))) (cons (list ct (car ls))
        (afterbearings (cdr ls) (+ ct 1))))
      (else (afterbearings (cdr ls) (+ ct 1))))))

;; aux de afterbearings
(define todosatoms?
  (lambda (l)
    (cond
      ((null? l) #t)
      ((atom? (car l)) (todosatoms? (cdr l)))
      (else #f))))

;; devuleve lista de oraciones que tienen por lo menos un
;; vector, con el numero de oracion
(define afterbearingsvec
  (lambda (ls ct)
    (cond
      ((null? ls) '())
      ((and
        (list? (car ls))
        (todoslistabuena? (car ls)))
       (cons (list ct (car ls))
        (afterbearingsvec (cdr ls) (+ ct 1))))
      (else (afterbearingsvec (cdr ls)
        (+ ct 1))))))

;; aux de afterbearingsvec
(define todoslistabuena?
  (lambda (l)
    (cond
      ((null? l) #f)
      ((atom? (car l)) (todoslistabuena? (cdr l)))
      ((vector? (caar l)) #t)
      (else (todoslistabuena? (cdr l))))))

;; retorna verdad si todos los elementos de la lista son
;; listas tambien
(define my-sontodoslistas?
  (lambda (ls)
    (cond
      ((null? ls) #t)
```

```
      ((list? (car ls)) (my-sontodoslistas? (cdr ls)))
      (else #f))))

;; ver si el resultado de Getyourbearings es
;; completamente koscher
(define koscherbearings?
  (lambda (ls)
    (if (not (list? (car ls)))
        #f
        (koscherbearings2? ls))))

;; auxiliar of koscherbearings?
(define koscherbearings2?
  (lambda (ls)
    (cond
      ((null? ls) #t)
      ((or
        (lista-atomica? (car ls))
        (lista-mixta? (car ls))
        (lista-vectores? (car ls)))
       (koscherbearings2? (cdr ls)))
      (else #f))))

;; auxiliar de koscherbearings?
(define lista-atomica?
  (lambda (l)
    (todosatoms? l)))

;; auxiliar de koscherbearings?
(define lista-mixta?
  (lambda (l)
    (cond
      ((null? l) #t)
      ((or
        (atom? (car l))
        (and
          (list? (car l))
          (vector? (caar l))
          (atom? (car (cdr (car l))))))
       (lista-mixta? (cdr l)))
      (else #f))))

;; auxiliar de koscherbearings?
(define lista-vectores?
  (lambda (l)
    (cond
      ((null? l) #t)
      ((and
        (list? (car l))
        (vector? (caar l))
        (atom? (car (cdr (car l)))))
       (lista-vectores? (cdr l)))
      (else #f))))
```

C.3 HAC Code

Note: some fragments of the code with perceived implementation/technique value, have been omitted as they might be commercialised in the immediate future.

```
;;
;;=====================================================================
;; LOAD at once CALCULUS.ss and other utilities
(load "/Users/orestesappel/Desktop/SchemeLibrary/Code/sorcalcv3.ss")
;;=====================================================================
;;================================
;;=== Fuzzy Vesion of SORcalc ===
;;================================
;;
;;=======================================
;;=== First Step: Validate LEXicons ======
;;=======================================
;;(read-file "/Users/orestesappel/Desktop/SchemeLibrary/Data/mylexobjsv7.ss")
;;(set! primerlex readbuffer)
;;(set! alterlex '())
;;
;; GLOBAL MAXIMORUM Variable
(set! maximorumx -1.0)
;;
;;=== To modufy POS, OBJ & NEG Scores
;;---------------------------------------------------------------------
;; Orestes Appel - LexAlterEditor
;;---------------------------------------------------------------------
;;=============================================
;;=== Initial CALL TO THE SYSTEM by the User ===
;;=============================================
(define lexedalter
  (lambda ()
    (begin
      (welcome)
      (procesadoralter primerlex 1))))
;;
;; initial header when entering the system
;;
(define welcomealter
  (lambda ()
    (display "-------------------------------------------------------")
    (newline)
    (display "---    Welcome to the Lexicon Editor System        ---")
    (newline)
    (display "-------------------------------------------------------")
    (newline)
    (display "A pair (Vector Orientation) will be displayed.
              Please enter the desired POS and NEG Orientation
              values separated by BLANK and hit ENTER.")
    (newline)
    (display "Enter the word END when done.")
    (newline)))
```

```
;;
;; initial dialog to decide the proper way of entering data
;;
(define procesadoralter
  (lambda (l count)
    (if (null? l)
        (elbanner)
        (begin
          (newline)
          (if (or
                (and (number? (getposscore (caar l)))
                  (> (getposscore (caar l)) 0.0))
                (and (number? (getnegscore (caar l)))
                  (> (getnegscore (caar l)) 0.0)))
              (begin
                (display (list count (car l)))
                (set! alterlex
                  (snoc (patch-no (car l)) alterlex))
                (procesadoralter (cdr l) (+ count 1)))
              (begin
                (display (list count (car l)))
                (newline)
                (display "Your␣selection␣for␣POS␣(enter␣END␣to␣end,
␣␣␣
```

```
                                        (and
                                          (number? sel2)
                                          (>= sel2 0.0)
                                          (<= sel2 1.0)
                                          (<= (+ sel sel2) 1.0))
                                          (begin
                                            (set! alterlex (snoc (patch-vec-alter
                                              (car l) sel sel2) alterlex))
                                            (newline)
                                            (procesadoralter (cdr l) (+ count 1)))
                                          (procesadoralter l count))))
                                    (procesadoralter l count))))))))))))))))


;; OLD Version: don't use it
(define procesadoralterzzz
  (lambda (l count)
    (if (null? l)
        (elbanner)
        (begin
          (newline)
          (display (list count (car l)))
          (newline)
          (display "Your selection for POS (enter END to end,
                    and Q for no change) ==> ")
          (let ((sel (read)))
            (begin
              (if (or
                    (equal? sel 'end)
                           (equal? sel 'END))
                  (elbannergb l)
                  (begin
                    (if (or
                          (equal? sel 'q)
                          (equal? sel 'Q))
                        (begin
                          (set! alterlex
                            (snoc (patch-no (car l)) alterlex))
                          (newline)
                          (procesadoralter (cdr l) (+ count 1)))
                        (begin
                          (if (number? sel)
                            (begin
                              (display sel)
                              (newline)
                              (display "Your selection for NEG ==> ")
                              (let ((sel2 (read)))
                                (if (number? sel2)
                                    (begin
                                      (display sel2)
                                      (set! alterlex
                                        (snoc (patch-vec-alter
                                        (car l) sel sel2) alterlex))
                                      (newline)
```

```
                                    (procesadoralter
                                      (cdr l) (+ count 1)))
                                  (procesadoralter l count))))
                               (procesadoralter l count))))))))))))


;; Patch vector appropiately with POS, NEG and OBJ Scores
(define patch-vec-alter
  (lambda (todo sop son)
    (list
      (list->vector (list (getword (car todo))
                          (getpos (car todo))
                          sop
                          son
                          (- 1.0 (+ sop son))
                          (getsor (car todo))
                          (getmaxdist (car todo))
                          (getmindist (car todo))
                          (getupdtindex (car todo))))
      (car (cdr todo)))))

;; called when no changes on vector are required
(define patch-no
  (lambda (ls)
    ls))

;;
;; Banner for application LEXedalter
(define elbanner
  (lambda ()
          (begin
            (newline)
            (display "----------------------------------------")
            (newline)
            (display "--- Thanks for using the system.
                          Have a nice day. ---")
            (newline)
            (display "----------------------------------------")
            (newline))))


;;
;; Banner for application LEXedalter
(define elbannergb
  (lambda (ele)
        (begin
          (newline)
          (display "----------------------------------------")
          (newline)
          (display "--- Thanks for using the system.
                          Have a nice day. ---")
          (newline)
          (display "----------------------------------------")
          (newline)
```

```
          (set! alterlex (append alterlex ele)))))

;; Validation-2 for good-structure of the Lexicon (goodpanalex? is Validation-1)
(define superpanalex?
  (lambda (lex)
    (cond
      ((null? lex) #t)
      ((and
         (and
           (number? (getposscore (caar lex)))
           (>= (getposscore (caar lex)) 0.0)
           (<= (getposscore (caar lex)) 1.0))
         (and
           (number? (getnegscore (caar lex)))
           (>= (getnegscore (caar lex)) 0.0)
           (<= (getnegscore (caar lex)) 1.0))
         (and
           (number? (getobjscore (caar lex)))
           (>= (getobjscore (caar lex)) 0.0)
           (<= (getobjscore (caar lex)) 1.0)))
       (superpanalex? (cdr lex)))
      (else #f))))


;; full health-check on Lexicons
(define lexhealthy?
  (lambda (lex)
    (and
      (goodpanalex? lex)
      (superpanalex? lex))))


;; calculate how many values are numbers given a lexicon,
;; a label 'POS (or NEG or OBJ)
;; and function (getposcore, getnegscore or getobjscore)
(define howmany-num
  (lambda (ls part myfun)
    (cond
      ((null? ls) 0)
      ((and
         (equal? (car (cdr (car ls))) part)
         (vector? (car (car ls)))
         (number? (myfun (car (car ls))))
         (number? (myfun (car (car ls)))))
       (+ 1 (howmany-num (cdr ls) part myfun)))
      (else (howmany-num (cdr ls) part myfun)))))

;; calculate SUM of values for a given laebl
;; (POS, NEG, OBJ) and a given getscore
;;  (getposscore, getnegscore, getobjscore)
(define howmany-num-suma
  (lambda (ls part myfun)
    (cond
      ((null? ls) 0)
```

```
      ((and
         (equal? (car (cdr (car ls))) part)
         (vector? (car (car ls)))
         (number? (getnegscore (car (car ls))))
         (number? (getobjscore (car (car ls))))
         (number? (getposscore (car (car ls))))
         (number? (myfun (car (car ls)))))
      (+ (myfun (car (car ls)))
         (howmany-num-suma (cdr ls) part myfun)))
      (else (howmany-num-suma
        (cdr ls) part myfun)))))


;; calcula MAX value for POS, NEG and OBJ scores
;; lex=lexicon; part='pos (or other label depending on
;; Subset); max=starting value (-1.0); and
;; myfun = getposscore, getnegscore or getobjscore
(define max-pos-how
  (lambda (lex part max myfun)
    (cond
      ((null? lex) max)
      ((> (myfun (caar lex)) max)
        (max-pos-how (cdr lex) part
          (myfun (caar lex)) myfun))
      (else (max-pos-how (cdr lex)
        part max myfun)))))

;; calcula MIN value for POS, NEG and OBJ scores
;; lex=lexicon; part='pos (or other label depending on
;; Subset); min=starting value (99.0); and
;; myfun = getposscore, getnegscore or getobjscore
(define min-pos-how
  (lambda (lex part min myfun)
    (cond
      ((null? lex) min)
      ((< (myfun (caar lex)) min)
        (min-pos-how (cdr lex) part (myfun (caar lex)) myfun))
      (else (min-pos-how (cdr lex) part min myfun)))))

;;===================================
;;=== END Validate LEXicons =========
;;===================================


;;=====================================================
;;=== Calculation for Membership Functions
;;      for  Fuzzy Techniques ===
;;=====================================================
;;
;;(define calc-mf
;;  (lambda (num)
;;    (cond
;;      ((< num 0.000) 'errlt0)
;;      ((and (>= num 0.000) (< num 0.225))
```

```
;;            (or (calc-mf-vneg num) (calc-mf-neg num)))
;;       ((and (>= num 0.225) (< num 0.450))
;;            (or (calc-mf-neg num) (calc-mf-obj num)))
;;       ((and (>= num 0.450) (< num 0.675))
;;            (or (calc-mf-obj num) (calc-mf-pos num)))
;;       ((and (>= num 0.675) (< num 0.900))
;;            (or (calc-mf-pos num) (calc-mf-vpos num)))
;;       ((>= num 0.900) (calc-mf-vpos num))
;;       ((> num 1.000) 'errgt1))))

;; Returns list of the form
;; (output-for-VNEG output-for-NEG output-for-OBJ
;; output-for-POS output-for-VPOS)
;; Calculate the MF value for a given entry (num)
(define calc-mf
  (lambda (num)
    (list
      (calc-mf-weak num)
      (calc-mf-mild num)
      (calc-mf-moderate num)
      (calc-mf-very num)
      (calc-mf-extreme num))))


;; MF Weak
(define calc-mf-weak
  (lambda (num)
    (cond
      ((and (>= num 0.000) (<= num 0.100)) 1.00)
      ((and (>= num 0.100) (<= num 0.225))
        (/ (- 0.225 num) (- 0.225 0.100)))
      ((= num 0.225) 0.00)
      (else 'nv-weak))))

;; MF Mild
(define calc-mf-mild
  (lambda (num)
    (cond
      ((< num 0.100) 'nv-mild)
      ((and (>= num 0.100) (<= num 0.225))
        (/ (- num 0.100) (- 0.225 0.100)))
      ((and (>= num 0.225) (<= num 0.325)) 1.00)
      ((and (>= num 0.325) (<= num 0.450))
        (/ (- 0.450 num) (- 0.450 0.325)))
      ((= num 0.450) 0.00)
      (else 'nv-mild))))

;; MF Moderate
(define calc-mf-moderate
  (lambda (num)
    (cond
      ((< num 0.325) 'nv-moderate)
      ((and (>= num 0.325) (<= num 0.450))
        (/ (- num 0.325) (- 0.450 0.325)))
```

```
      ((and (>= num 0.450) (<= num 0.550)) 1.00)
      ((and (>= num 0.550) (<= num 0.675))
        (/ (- 0.675 num) (- 0.675 0.550)))
      ((= num 0.675) 0.00)
      (else 'nv-moderate))))

;; MF Very
(define calc-mf-very
  (lambda (num)
    (cond
      ((< num 0.550) 'nv-very)
      ((and (>= num 0.550) (<= num 0.675))
        (/ (- num 0.550) (- 0.675 0.550)))
      ((and (>= num 0.675) (<= num 0.775)) 1.00)
      ((and (>= num 0.775) (<= num 0.900))
        (/ (- 0.900 num) (- 0.900 0.775)))
      (else 'nv-very))))

;; MF Extreme
(define calc-mf-extreme
  (lambda (num)
    (cond
      ((< num 0.775) 'nv-extreme)
      ((and (>= num 0.775) (<= num 0.900))
        (/ (- num 0.775) (- 0.900 0.775)))
      ((and (>= num 0.900) (<= num 1.00)) 1.00)
      (else 'nv-extreme))))

;; Returns list of the form (output-for-VNEG
;;   output-for-NEG output-for-OBJ
;;  output-for-POS output-for-VPOS)
;; Calculate the MF value for a given entry (num)
(define calc-mf-ample
  (lambda (num)
    (list
      (calc-mf-poor2 num)
      (calc-mf-slightly2 num)
      (calc-mf-moderate2 num)
      (calc-mf-very2 num)
      (calc-mf-most2 num))))


;; MF Weak
(define calc-mf-poor2
  (lambda (num)
    (cond
      ((and (>= num 0.000) (<= num 0.050)) 1.00)
      ((and (>= num 0.100) (<= num 0.150))
        (/ (- 0.150 num) (- 0.150 0.100)))
      ((= num 0.15) 0.00)
      (else 'nv-poor))))

;; MF Mild
(define calc-mf-slightly2
```

```
  (lambda (num)
    (cond
      ((< num 0.050) 'nv-slightly)
      ((and (>= num 0.100) (<= num 0.225))
        (/ (- num 0.100) (- 0.225 0.100)))
      ((and (>= num 0.225) (<= num 0.325)) 1.00)
      ((and (>= num 0.325) (<= num 0.450))
        (/ (- 0.450 num) (- 0.450 0.325)))
      ((= num 0.450) 0.00)
      (else 'nv-slightly))))

;; MF Moderate
(define calc-mf-moderate2
  (lambda (num)
    (cond
      ((< num 0.325) 'nv-moderate)
      ((and (>= num 0.325) (<= num 0.450))
        (/ (- num 0.325) (- 0.450 0.325)))
      ((and (>= num 0.450) (<= num 0.550)) 1.00)
      ((and (>= num 0.550) (<= num 0.675))
        (/ (- 0.675 num) (- 0.675 0.550)))
      ((= num 0.675) 0.00)
      (else 'nv-moderate))))

;; MF Very
(define calc-mf-very2
  (lambda (num)
    (cond
      ((< num 0.550) 'nv-very)
      ((and (>= num 0.550) (<= num 0.675))
        (/ (- num 0.550) (- 0.675 0.550)))
      ((and (>= num 0.675) (<= num 0.775)) 1.00)
      ((and (>= num 0.775) (<= num 0.900))
        (/ (- 0.900 num) (- 0.900 0.775)))
      (else 'nv-very))))

;; MF Extreme
(define calc-mf-most2
  (lambda (num)
    (cond
      ((< num 0.775) 'nv-most)
      ((and (>= num 0.775) (<= num 0.900))
        (/ (- num 0.775) (- 0.900 0.775)))
      ((and (>= num 0.900) (<= num 1.00)) 1.00)
      (else 'nv-most))))


;; Global variable ELNEGSET contains outcome of
;;   GETYOURBEARINGS for NEG set
;; Global variable ELPOSSET contains outcome of
;;   GETYOURBEARINGS for POS set
;;
;;=============================================
;;=== FIltra resultados de getyour-bearings ====
```

```
;;=============================================
;; delete from the output of GETYOURBEARINGS words
;;   with no lexicon match
(define losbuenosy
  (lambda (ls)
    (if (null? ls)
        '()
        (cons (deja-losbuenosy (car ls))
          (losbuenosy (cdr ls))))))

;; auxiliar de LOSBUENOSy
(define deja-losbuenosy
  (lambda (lst)
    (cond
      ((null? lst) '())
      ((atom? (car lst))
        (deja-losbuenosy (cdr lst)))
      (else (cons (car lst)
        (deja-losbuenosy (cdr lst)))))))

;; replace in output of LOSBUENOSy los vacios por NOSOR
;;   y agrega el numero de oracion
(define filtra-losbuenosy
  (lambda (ls ct)
    (cond
      ((null? ls) '())
      ((equal? (car ls) '())
        (cons (list ct 'nosor) (filtra-losbuenosy
        (cdr ls) (+ ct 1))))
      (else
        (cons (list ct (car ls)) (filtra-losbuenosy
        (cdr ls) (+ ct 1)))))))

;;=================================================
;;=== END de filtra resultados de GETYOURBEARINGS ===
;;=================================================
;;
;; produce lista de forma ( (num ((MaxPosValue
;;      MinPosValue NumOfPoss) (MaxNegValue MinNegValue
;;      NumofNegs) (MaxObjValue MinObjValue
;;      NumofObjs) ))
;; ... los siguientes para cada oracion NUM )
;; Input = Salida de FILTRA-LOSBUENOSy
(define paso2sorcalc2
  (lambda (ls)
    (cond
      ((null? ls) '())
      (else (cons (list (caar ls)
        (paso2sorcalc2-aux (car (cdr (car ls)))))
        (paso2sorcalc2 (cdr ls)))))))

;; auxiliar de PASO2SORCALC2
(define paso2sorcalc2-aux
  (lambda (ls)
```

```
    (cond
      ((null? ls) '())
      ((equal? ls 'nosor) 'nosor)
      (else (list (list (losmaxxy ls 'pos)
        (losminny ls 'pos)
        (cuantosmaxxy ls 0))
                  (list (losmaxxy ls 'neg)
                    (losminny ls 'neg)
                    (cuantosminny ls 0))
                  (list (losmaxxy ls 'obj)
                    (losminny ls 'obj)
                    (cuantosmmobj ls 0)))))))

;; cuenta cuanta matches de lexicon por oracion son POS
(define cuantosmaxxy
  (lambda (l conta)
    (if (null? l)
        conta
       (if (equal? (car (cdr (car l))) 'pos)
            (cuantosmaxxy (cdr l) (+ conta 1))
            (cuantosmaxxy (cdr l) conta)))))

;; cuenta cuanta matches de lexicon por oracion son NEG
(define cuantosminny
  (lambda (l conta)
    (if (null? l)
        conta
        (if (equal? (car (cdr (car l))) 'neg)
            (cuantosminny (cdr l) (+ conta 1))
            (cuantosminny (cdr l) conta)))))

;; cuenta cuanta matches de lexicon por oracion son OBJ
(define cuantosmmobj
  (lambda (l conta)
    (if (null? l)
        conta
        (if (equal? (car (cdr (car l))) 'obj)
            (cuantosmmobj (cdr l) (+ conta 1))
            (cuantosmmobj (cdr l) conta)))))

;; calcula el MAXIMUM among los PositiveScores para
;;   las palabras
;; de tipo ETIQ=POS, NEG or OBJ
(define losmaxxy
  (lambda (l etiq)
    (bestia-max (losmaxxy-aux l etiq) -1.00)))

;; auxiliar of LOSMAXXY
(define losmaxxy-aux
  (lambda (l etiq)
    (cond
      ((null? l) '())
      ((equal? (car (cdr (car l))) etiq)
        (cons (positron (caar l))
```

```
        (losmaxxy-aux (cdr l) etiq)))
      (else (losmaxxy-aux (cdr l) etiq)))))

;; calcula el MAXIMUM among los NegativeScores
;;   para las palabras
;; de tipo ETIQ=POS, NEG or OBJ
(define losminny
  (lambda (l etiq)
    (bestia-max (losminny-aux l etiq) -1.00)))

;; auxuliar de LOSMINNY
(define losminny-aux
  (lambda (l etiq)
    (cond
      ((null? l) '())
      ((equal? (car (cdr (car l))) etiq)
        (cons (negatron (caar l))
        (losminny-aux (cdr l) etiq)))
      (else (losminny-aux (cdr l) etiq)))))

;; calcula el maximo en una lista de rational nums
(define bestia-max
  (lambda (l max)
    (cond
      ((null? l) max)
      ((> (car l) max)
        (bestia-max (cdr l) (car l)))
      (else (bestia-max (cdr l) max)))))

;; ruturns el POSITIVE score of a given word
(define positron
  (lambda (v)
    (vector-ref v 2)))

;; returns el NEGATIVE score of a given word
(define negatron
  (lambda (v)
    (vector-ref v 3)))

;; retorna el primer elemento de la lista
;; (MAXPOSSCORE MAXNEGSCORE NUMOFELEMENTSOFTHISTYPE)
(define get-maxpscore
  (lambda (l)
    (car l)))

;; retorna el segundo elemento de la lista
;; (MAXPOSSCORE MAXNEGSCORE NUMOFELEMENTSOFTHISTYPE)
(define get-maxnscore
  (lambda (l)
    (car (cdr l))))

;; retorna el tercer elemento de la lista
;; (MAXPOSSCORE MAXNEGSCORE NUMOFELEMENTSOFTHISTYPE)
(define get-numele
```

```
  (lambda (l)
    (car (cdr (cdr l)))))


;;=======================================================
;;=== Starts SOR Calculation using Fuzzy Techniques ===
;;=======================================================
;; Main call de CALC-FUZZY
;;
;;(set! eluno (losbuenosy ls))
;;(set! eldos (filtra-losbuenosy eluno 1))
;;(set! eltres (paso2sorcalc2 eldos))
;;=====================================================================
;; 1. Global variables ELSIETEP and ELSIETEN contain list of the form: =
;; ( (1 pos (pos 0.5) moderate) ... )                                  ===
;; 2. Global variables LISTOPACOMPAP and LISTOPACOMPAN contains
;;                                                  list of the form: ===
;; ( (pos 1.0) (neg 0.77) ... (nosor nosor) )
;;=====================================================================
;;
(define calc-fuzzy
  (lambda ()
    (begin
      (newline)
      (display "=====================================================================")
      (newline)
      (display "BEGINNING Results using Hybrid Method HAC.")
      (newline)
      (set! elunop (losbuenosy elposset))
      (set! eldosp (filtra-losbuenosy elunop 1))
      (set! eltresp (paso2sorcalc2 eldosp))
      (set! elcuatrop (calcula-true eltresp))
      (set! elcincop (filtra-actuario elcuatrop))
      (set! elseisp (contador-x elcincop))
      (set! elsietep
        (joint-fuzzy-calc-trans resulposz elcuatrop))
      (set! listopacompap (saca-val-compara elsietep))
      (set! elunon (losbuenosy elnegset))
      (set! eldosn (filtra-losbuenosy elunon 1))
      (set! eltresn (paso2sorcalc2 eldosn))
      (set! elcuatron (calcula-true eltresn))
      (set! elcincon (filtra-actuario elcuatron))
      (set! elseisn (contador-x elcincon))
      (set! elsieten
        (joint-fuzzy-calc-trans resulnegz elcuatron))
      (set! listopacompan (saca-val-compara elsieten))
      (stats-finas-fuzzy elsieten elsietep)
      (newline)
      (newline)
      (display "END Results using Hybrid Method HAC.")
      (newline)
      (display "=====================================================================")
      (newline)
      (newline)
```

```
      )))


;; actual calculation of SORs
(define calcula-true
  (lambda (ls)
    (cond
      ((null? ls) '())
      ((equal? (car (cdr (car ls))) 'nosor)
        (cons (car ls)
        (calcula-true (cdr ls))))
      (else (cons (list (caar ls)
        (calcula-true-aux (car (cdr (car ls)))))
      (calcula-true (cdr ls)))))))

(define calcula-true-aux
  (lambda (l)
    (cond
      ((null? l) '())
      (else (calcula-true-aux-z l)))))

(define calcula-true-aux-z
  (lambda (l)
    (cond
      ((and
         (= (get-numele (car l)) 0)
         (= (get-numele (car (cdr l))) 0))
      (list (what-todow-obj
          (car (cdr (cdr l))) 0.5))
      ((and
         (= (get-numele (car l)) 0)
         (> (get-numele (car (cdr l))) 0)) (list 'neg
         (get-maxnscore (car (cdr l)))))
      ((and
         (> (get-numele (car l)) 0)
         (= (get-numele (car (cdr l))) 0))
      (list 'pos (get-maxpscore (car l))))
      ((> (get-numele (car l)) (get-numele (car (cdr l))))
        (list 'pos (get-maxpscore (car l))))
      ((< (get-numele (car l)) (get-numele (car (cdr l))))
        (list 'neg (get-maxnscore (car (cdr l)))))
      ((= (get-numele (car l)) (get-numele (car (cdr l))))
        (if (> (get-maxpscore
          (car l)) (get-maxnscore (car (cdr l))))
        (list 'pos (get-maxpscore (car l)))
        (list 'neg (get-maxnscore (car (cdr l))))))
      (else 'workingonit))))


;; what to do with OBJs
(define what-todow-obj
  (lambda (l)
    (cond
      ((= (get-numele l) 0)
```

```
        (aleatorio 10))
      ((> (get-maxpscore l)
        (get-maxnscore l)) 'pos)
      ((< (get-maxpscore l)
        (get-maxnscore l)) 'neg)
      (else (aleatorio 10)))))


;; random option when no more choices are left
(define aleatorio
  (lambda (seedx)
    (if (even? (random seedx))
      'pos
      'neg)))


;; process NOSORs
(define filtra-actuario
  (lambda (ls)
    (cond
      ((null? ls) '())
      ((equal? (car (cdr (car ls))) 'nosor)
        (cons 'nosor (filtra-actuario
          (cdr ls))))
      (else (cons (caar (cdr (car ls)))
        (filtra-actuario (cdr ls)))))))

;; count the proper occurences of POS, NEG,
;;    OBJ and NOSOR
(define contador-x
  (lambda (ls)
    (list (cuentamecomo ls 'pos)
      (cuentamecomo ls 'neg)
      (cuentamecomo ls 'obj)
      (cuentamecomo ls 'nosor))))


;; actual counting function. Auxiliar to contador-x
(define cuentamecomo
  (lambda (ls etiq)
    (cond
      ((null? ls) 0)
      ((equal? (car ls) etiq)
        (+ 1 (cuentamecomo (cdr ls) etiq)))
      (else (cuentamecomo (cdr ls) etiq)))))


;;=============================================================================
;;=== Generate intensity of polarity using:                                 ===
;;=== Variable global RESULPOSZ and RESULNEGZ, outputs of CALC-SIMPLE.      ===
;;=== Variable global ELCINCOPOS and ELCINCONEG, outputs of CALC-FUZZY,     ===
;;=== each of the form: ((1 (neg 0.3)) ... (Num_Sentence
;;                                    (neg/pos MaxValueFor Polarity))) ===
;;=============================================================================
```

```
;;
(define joint-fuzzy-calc-trans
  (lambda (lsimple lcomplex)
    (cond
      ((null? lsimple) '())
      ((equal? (car lsimple) 'nosor)
        (cons (list (caar lcomplex) 'nosor)
        (joint-fuzzy-calc-trans
          (cdr lsimple) (cdr lcomplex))))
      (else (cons
        (list (caar lcomplex) (car lsimple)
        (car (cdr (car lcomplex)))
        (decode-z (p-square (c-square (calc-mf-ample
            (car (cdr (car (cdr (car lcomplex)))))))) 1 -1.0 1)) maximorumx)
        (joint-fuzzy-calc-trans (cdr lsimple) (cdr lcomplex)))))))

;; manages the non-numeric values of the output, like nv-most,
;;   nv-poor and the rest
(define c-square
  (lambda (ls)
    (cond
      ((null? ls) '())
      ((not (number? (car ls))) (cons -99 (c-square (cdr ls))))
      (else (cons (car ls) (c-square (cdr ls)))))))


;; calculate the MAX of the provided outcome values for the 5 MFs
(define p-square
  (lambda (ls cnt max indice)
    (cond
      ((null? ls) (begin (set! maximorumx max) indice))
      ((= (car ls) -99) (p-square (cdr ls)
        (+ cnt 1) max indice))
      ((>= (car ls) max) (p-square (cdr ls)
        (+ cnt 1) (car ls) cnt))
      (else (p-square (cdr ls) (+ cnt 1)
        max indice)))))


;; decode the output of p-square
(define decode-z
  (lambda (num)
    (cond
      ((= num 1) 'poor)
      ((= num 2) 'slightly)
      ((= num 3) 'moderate)
      ((= num 4) 'very)
      ((= num 5) 'most)
      (else 'nointensity))))


;;
;;=====================================================================
;; lsneg = ELSIETEN; lspos = ELSIETEP
```

```
(define stats-finas-fuzzy
  (lambda (lsneg lspos)
    (begin
      (newline)
      (display "OUTPUT FORMAT for POSs Dataset:
        (NumOfPOOR NumOfSLIGHTLY NumOfModerate NumOfVery
          NumOfMost NumOfNOSOR NumOfNegative")
      (newline)
      (display (list (mira-z-poss lspos 'poor)
        (mira-z-poss lspos 'slightly)
        (mira-z-poss lspos 'moderate)
        (mira-z-poss lspos 'very)
        (mira-z-poss lspos 'most)
        (mira-posneg-nosor lspos)
        (mira-zpecial-negs lspos)))
      (newline)
      (display "OUTPUT FORMAT for NEGs Dataset:
        (NumOfPOOR NumOfSLIGHTLY NumOfModerate NumOfVery
          NumOfMost NumOfNOSOR NumOfPositive")
      (newline)
      (display (list (mira-z-negs lsneg 'poor)
        (mira-z-negs lsneg 'slightly)
        (mira-z-negs lsneg 'moderate)
        (mira-z-negs lsneg 'very)
        (mira-z-negs lsneg 'most)
        (mira-posneg-nosor lsneg)
        (mira-zpecial-poss lsneg)))
      (newline))))

;; yields number of POS classifications with intensity INTEN
(define mira-z-poss
  (lambda (ls inten)
    (cond
      ((null? ls) 0)
      ((and
         (equal? (car (cdr (car ls))) 'pos)
         (equal? (car (cdddr (car ls))) inten))
      (+ 1 (mira-z-poss (cdr ls) inten)))
      (else (mira-z-poss (cdr ls) inten)))))

;; yields number of NEG classifications with intensity INTEN
(define mira-z-negs
  (lambda (ls inten)
    (cond
      ((null? ls) 0)
      ((and
         (equal? (car (cdr (car ls))) 'neg)
         (equal? (car (cdddr (car ls))) inten))
      (+ 1 (mira-z-negs (cdr ls) inten)))
      (else (mira-z-negs (cdr ls) inten)))))

;; yields number of NOSOR classifications, for POSs and NEGs
(define mira-posneg-nosor
  (lambda (ls)
```

```
    (cond
      ((null? ls) 0)
      ((equal? (car (cdr (car ls))) 'nosor)
        (+ 1 (mira-posneg-nosor (cdr ls))))
      (else (mira-posneg-nosor (cdr ls))))))

;; Generates lista de la forma ( (pos/neg   Fuzzy-Value
;;  Fuzzy-Value MAPPED) or (nosor nosor) ... ) of LENGTH = 3 .
;; Results deposited in global variables LISTOPACOMPAP
;;  and LISTOPACOMPAN
(define saca-val-compara
  (lambda (ls)
    (cond
      ((null? ls) '())
      ((equal? (car (cdr (car ls))) 'nosor)
        (cons (list 'nosor 'nosor)
        (saca-val-compara (cdr ls))))
      (else (cons (list (car (cdr (car ls)))
        (car (cddddr (car ls)))
        (if (equal? (car (cdr (car ls))) 'neg)
          (map-cambio 0.0 1.0 0.0 0.5000 (car (cddddr (car ls))))
          (map-cambio 0.0 1.0 0.5001 1.0 (car (cddddr (car ls))))))
      (saca-val-compara (cdr ls)))))))


;;
;; Special to count POSs in List of Negative labels
;;
(define mira-zpecial-poss
  (lambda (ls)
    (cond
      ((null? ls) 0)
      ((equal? (car (cdr (car ls))) 'pos)
        (+ 1 (mira-zpecial-poss (cdr ls))))
      (else (mira-zpecial-poss (cdr ls))))))

;;
;; Special to count NEGs in List of Positive labels
;;
(define mira-zpecial-negs
  (lambda (ls)
    (cond
      ((null? ls) 0)
      ((equal? (car (cdr (car ls))) 'neg)
        (+ 1 (mira-zpecial-negs (cdr ls))))
      (else (mira-zpecial-negs (cdr ls))))))



;; call HSC, HAC and HACA: latol = tolerance = 0.5000
(define thefull-calc
  (lambda (latol)
    (begin
      (calc-sor-allset-simple)
```

```
    (calc-fuzzy)
    (calc-iowa-stuff latol))))
```

C.4 HACACU & HACACO Code

Note: some fragments of the code with perceived implementation/technique value, have been omitted as they might be commercialised.

```
;;
;;==============================
;; LOAD Files for IOWA ==========
;;==============================
(read-file "/Users/orestesappel/Desktop/SchemeLibrary/DataPython/
  NaiveBayes/negativecasesnb.txt")
(set! nbnegx readbuffer)
(read-file "/Users/orestesappel/Desktop/SchemeLibrary/DataPython/
  NaiveBayes/positivecasesnb.txt")
(set! nbposx readbuffer)
(read-file "/Users/orestesappel/Desktop/SchemeLibrary/DataPython/
  MaxEntropy/negativecasesme.txt")
(set! menegx readbuffer)
(read-file "/Users/orestesappel/Desktop/SchemeLibrary/DataPython/
  MaxEntropy/positivecasesme.txt")
(set! meposx readbuffer)
;;==============================
;;============================================================
;;=== Segment for calculating SOR using IOWA and results
;;       of 3 methods ===
;;============================================================
;;
;;===========================================
;;=== Sentences choosing and identification ===
;;===========================================
;; generate list of 3 sentences numbers meeting given criteria
;; Call like (provide-sentnum-label ls 1 0 'slightly),
;;   where ls = ELSIETEN and ELSIETEP
(define provide-sentnum-label
  (lambda (ls cont flaggy label)
    (cond
      ((or
         (null? ls)
         (= flaggy 3)) '())
      ((equal? (car (car ls)) 'nosor)
        (provide-sentnum-label
          (cdr ls) (+ cont 1) label))
      ((equal? (car (cdddr (car ls))) label)
        (cons cont (provide-sentnum-label (cdr ls)
          (+ cont 1) (+ flaggy 1) label)))
      (else (provide-sentnum-label (cdr ls)
        (+ cont 1) flaggy label)))))

;; Retorna la oracion marcada con el numero NUMSENT
;; negorigs AND posorigs contain the original sentences
(define pasamela-y
```

```
  (lambda (ls sentnum cont)
    (cond
      ((null? ls) '())
      ((= sentnum cont) (car ls))
      (else (pasamela-y (cdr ls)
        sentnum (+ cont 1))))))

;; CONVERSION in Range (Linear)
;;Given two ranges, [a1,a2] and [b1,b2]; then a value
;; s in range [a1,a2]
;;is linearly mapped to a value t in range [b1,b2]
;;when: t = b1 + {(s - a1)(b2 - b1) \over (a2 - a1)}
;; Ejemplos:
;; (map-cambio 0.0 1.0 0.0 0.49 0.3) PRODUCE 0.147
;; (map-cambio 0.0 1.0 0.50 1.0 0.3) PRODUCE 0.650
;;
(define map-cambio
  (lambda (a1 a2 b1 b2 s)
    (+ b1
     (/ (* (- s a1)
           (- b2 b1))
        (- a2 a1)))))

(define map-nb-pos
  (lambda (val)
    (map-cambio 0.50000000 0.99999852 0.0 1.0 val)))

(define map-me-pos
  (lambda (val)
    (map-cambio 0.50000000451 0.500168230 0.0 1.0 val)))

(define map-nb-neg
  (lambda (val)
    (map-cambio 0.50000000 0.999987442 0.0 1.0 val)))

(define map-me-neg
  (lambda (val)
    (map-cambio 0.50000000459 0.500141235 0.0 1.0 val)))

;;
;; BUILD lista para suplir a IOWA Process
;; Input: Results-of-NB Results-of-ME listopacompan/p
;; Output: ( ((0.9390256821983421 pos)
;;         (0.5000406225810768 pos) (1.0 neg))
;;          ((0.860015516808406 pos)
;;         (0.500025678163244 pos) (nosor nosor))
;;            ((0.9790256821983421 pos)
;;         (0.5100040225810768 pos) (.95 neg)) )
;; For NEGs
(define buildit4iowa-neg
  (lambda (lnb lme lore)
    (if
      (null? lnb)
      '()
```

```
      (cons (process4iowa-neg
        (car lnb) (car lme) (car lore))
        (buildit4iowa-neg (cdr lnb)
          (cdr lme) (cdr lore))))))

;; For POSs
(define buildit4iowa-pos
  (lambda (lnb lme lore)
    (if
      (null? lnb)
      '()
      (cons (process4iowa-pos
        (car lnb) (car lme) (car lore))
        (buildit4iowa-pos
          (cdr lnb) (cdr lme) (cdr lore))))))


;; FORMATOS of Input
;; NB = (512  0.50000420247266397   neg   1)
;; me = (101  0.9249393530588206   pos   1)
;; ore = (pos 0.5999999999999996 0.800039)
;; OUTPUT = ( (0.50000 neg) (0.9249 pos)
;;                          (0.5999 pos) )
;; auxiliar to buildit4iowa
(define process4iowa-neg
  (lambda (nb me oreval)
    (list (list (map-nb-neg (car (cdr nb)))
      (car (cddr nb)))
          (list (map-me-neg (car (cdr me)))
            (car (cddr me)))
          (list (car (cdr oreval))
            (car oreval)))))

(define process4iowa-pos
  (lambda (nb me oreval)
    (list (list (map-nb-pos (car (cdr nb)))
      (car (cddr nb)))
          (list (map-me-pos (car (cdr me)))
            (car (cddr me)))
          (list (car (cdr oreval))
            (car oreval)))))


;;
;; replace 'nosors in output from buildit4iowa
;;
(define premasajea-before-iowa
  (lambda (ls)
    (cond
      ((null? ls) '())
      ((equal? (caar (cddr (car ls))) 'nosor)
        (cons (list (caar ls) (car (cdr (car ls)))
          (car (cdr (car ls))))
              (premasajea-before-iowa (cdr ls))))
```

```
      (else (cons (car ls)
        (premasajea-before-iowa (cdr ls)))))))

;; build the list to be passed to IOWA
(define extractx-vals-iowa
  (lambda (ls)
    (if (null? ls)
        '()
       (cons (extractx-iowax (car ls))
        (extractx-vals-iowa (cdr ls))))))

;; auxiliar to extract-vals-iowa
;; returns the list for IOWA sorted from less to more
(define extractx-iowax
  (lambda (ls)
    (list (caar ls) (caar (cdr ls))
      (caar (cddr ls)))))
;;    (list-sort < (list (caar ls) (caar (cdr ls))
;;     (caar (cddr ls))))))


;;=========================================
;;=== Calc for IOWA (ACTUAL) ==============
;;=========================================
;; Salidas of sor-calc-SIMPLE:
;; * resulposz: resultados de positive set.
;;   Format: ( neg pos ... neg)
;; * resulnegz: resultados de negative set.
;;   Format: ( neg pos ... pos)
;; paiowapos2: resultados POS listos para IOWA analysis.
;;   Format: (0.011 0.03  0.79),
;;     ordered already from lesser to greater.
;; paiowaneg2: resultados NEG listos para IOWA analysis.
;;   Format: same as above.
;;
;; Parameter to calc-iowa-stuff is TOLERANCE.
;;   Call with Tolerance = 0.500000.
(define calc-iowa-stuff
  (lambda (tolly)
    (begin
      (set! paiowapos1 (premasajea-before-iowa
        (buildit4iowa-pos nbposx meposx listopacompap)))
      (set! paiowapos2 (extractx-vals-iowa paiowapos1))
      (set! paiowaneg1 (premasajea-before-iowa
        (buildit4iowa-neg nbnegx menegx listopacompan)))
      (set! paiowaneg2 (extractx-vals-iowa paiowaneg1))
      (set! tolerance tolly)
      (set! veci '())
      (set! iowaposs '())
      (set! supveci '())
      (set! orderednewsupvec '())
      (set! aggrega 0.0)
      (set! iowaposs (iowafunlistpos paiowapos2))
      (set! veci '())
```

```
      (set! iowanegs '())
      (set! supveci '())
      (set! orderednewsupvec '())
      (set! aggrega 0.00)
      (set! iowanegs (iowafunlistneg paiowaneg2))
      (set! negnewt (proiowaworld paiowaneg1))
      (set! posnewt (proiowaworld paiowapos1))
      (newline)
      (display "============================================================")
      (newline)
      (display "BEGINNING Results Aggregation using IOWA
                with Hybrid Method HACA.")
      (newline)
      (newline)
      (jclassy iowanegs iowaposs)
      (newline)
      (newline)
      (display "END Results Aggregation using IOWA
                with Hybrid Method HACA.")
      (newline)
      (display "============================================================")
      (newline)
      (newline)
      (display "============================================================")
      (newline)
      (display "BEGINNING Results Aggregation using
                VOTING and AVERAGING instead of OWA Operators.")
      (newline)
      (dalesiniowa posnewt negnewt)
      (manipulaprom (damepromediogamma paiowapos2)
        (damepromediogamma paiowaneg2))
      (newline)      (newline)
      (display "END Results Aggregation using VOTING and AVERAGING
                instead of OWA Operators.")
      (newline)
      (display "============================================================")
      (newline)
      (newline)
      (newline)
      )))

;;(define veci '())
;;(define supveci '())
;;(define orderednewsupvec '())

;;
;; apply IOWA parameters to list of values of type
;;   ( (val1 val2 val3) ... (val1 val2 val3) )
(define iowafunlistpos
  (lambda (lst)
    (if (null? lst)
        '()
        (cons (iowafunkpos (car lst))
          (iowafunlistpos (cdr lst))))))
```

```
(define iowafunlistneg
  (lambda (lst)
    (if (null? lst)
        '()
        (cons (iowafunkneg (car lst))
          (iowafunlistneg (cdr lst))))))

;; apply IOWA parameters to one item of
;;    type (val1 val2 val3)
(define iowafunkpos
  (lambda (preveci)
  (begin
    (set! supveci (gen-sup-vector preveci tolerance 1))
    (set! veci (filter-thelval
      (mylist-sort-pair-1st
        (mylist-sort-pair-2nd
          (induced-sort-pairs supveci preveci)))))
    (set! orderednewsupvec
      (newsupvec
        (p1divp2
          (qtins (x-is-tofn (tis supveci)
            (length (tis supveci))))
          (sumqtins (qtins (x-is-tofn
            (tis supveci) (length (tis supveci))))))))
      (multmyvecs orderednewsupvec veci)
)))


;; apply IOWA parameters to one item of type (val1 val2 val3)
(define iowafunkneg
  (lambda (preveci)
  (begin
    (set! supveci (gen-sup-vector preveci tolerance 1))
    (set! veci (filter-thelval
      (mylist-sort-pair-1st
        (mylist-sort-pair-2nd
          (induced-sort-pairs supveci preveci)))))
    (set! orderednewsupvec
      (newsupvec
        (p1divp2
          (qtins (x-is-tofn (tis supveci)
            (length (tis supveci))))
          (sumqtins (qtins (x-is-tofn (tis supveci)
            (length (tis supveci)))))))
    (multmyvecs orderednewsupvec veci)
)))


;; is X a NAN?
(define isnan?
  (lambda (x)
    (not (= x x))))
```

```
;; cuantos NAN hay en una lista fr numeros
(define cuantosnan
  (lambda (ls)
    (cond
      ((null? ls) 0)
      ((isnan? (car ls))
        (+ 1 (cuantosnan (cdr ls))))
      (else (cuantosnan (cdr ls))))))


(define cuantos<=>xy
  (lambda (l x y)
    (cond
      ((null? l) 0)
      ((and
         (>= (car l) x)
         (< (car l) y))
      (+ 1 (cuantos<=>xy (cdr l) x y)))
      (else (cuantos<=>xy (cdr l) x y)))))


(define jclassy
  (lambda (lnegs lposs)
    (begin
      (let ((res (jclassy-aux lnegs lposs)))
        (newline)
        (newline)
        (newline)
        (display "POS Set results as COUNT of Classes from
                  POOR to MOST (using HACA): ")
        (display (car (cdr res)))
        (newline)
        (newline)
        (display "Results for POS Dataset [Format: (TP FP)]: ")
        (display (list (sumaweirdo (car (cdr res)))
          (car (car (cdr res)))))
        (newline)
        (if (not (= (+ (sumaweirdo (car (cdr res)))
               (car (car (cdr res)))) (length paiowapos1)))
            (begin (display "Potential OBJ sentences = ")
              (display (- (length paiowapos1)
                (+ (sumaweirdo (car (cdr res)))
                  (car (car (cdr res))))))) (newline))
            (begin (display "Potential OBJ sentences = 0")
              (newline)))
        (newline)
        (display "NEG Set results as COUNT of Classes from
                  POOR to MOST (using HACA): ")
        (display (car res))
        (newline)
        (newline)
        (display "Results for NEG Dataset [Format: (TN FN)]: ")
        (display (list (sumaweirdo (car res)) (caar res)))
```

```
        (newline)
        (if (not (= (+ (sumaweirdo (car res))
          (caar res)) (length paiowaneg1)))
            (begin (display "Potential OBJ sentences = ")
              (display (- (length paiowaneg1)
                (+ (sumaweirdo (car res)) (caar res)))) (newline))
            (begin (display "Potential OBJ sentences = 0")
              (newline)))
        (newline)
        (newline)
        (newline)))))


(define jclassy-aux
  (lambda (ln lp)
    (list (list (cuantos<=>xy ln 0.000 0.100)
                (cuantos<=>xy ln 0.150 0.350)
                (cuantos<=>xy ln 0.350 0.750)
                (cuantos<=>xy ln 0.750 0.950)
                (cuantos<=>xy ln 0.950 1.001))
          (list (cuantos<=>xy lp 0.000 0.150)
                (cuantos<=>xy lp 0.150 0.350)
                (cuantos<=>xy lp 0.350 0.750)
                (cuantos<=>xy lp 0.750 0.950)
                (cuantos<=>xy lp 0.950 1.001)))))

;; suma todos menos POOR
(define sumaweirdo
  (lambda (l5)
    (+ (+ (car (cdr l5)) (car (cddr l5)))
      (+ (car (cdddr l5)) (car (cddddr l5))))))


(define proiowaworld
  (lambda (ls)
    (if (null? ls)
        '()
        (cons (proiowaworld-aux (car ls))
          (proiowaworld (cdr ls))))))

(define proiowaworld-aux
  (lambda (l)
    (mylookupoa (car (list-sort > (list (caar l)
      (caar (cdr l)) (caar (cddr l))))) l)))

(define mylookupoa
  (lambda (val lt)
    (cond
      ((null? lt) '())
      ((= (caar lt) val) (car (cdr (car lt))))
      (else (mylookupoa val (cdr lt))))))

;;add those labelled as POS
(define consigue-alpha
```

```
  (lambda (l)
    (cond
      ((null? l) 0)
      ((equal? (car l) 'pos)
        (+ 1 (consigue-alpha (cdr l))))
      (else (consigue-alpha (cdr l)))))

    ;; add those labelled as NEG
(define consigue-beta
  (lambda (l)
    (cond
      ((null? l) 0)
      ((equal? (car l) 'neg)
        (+ 1 (consigue-beta (cdr l))))
      (else (consigue-beta (cdr l)))))


(define dalesiniowa
  (lambda (lp ln)
    (begin
      (newline)
      (display "For POS Dataset (without IOWA
                and using Voting), list of the form (TP FN): ")
      (display (list (consigue-alpha lp) (consigue-beta lp)))
      (newline)
      (display "For NEG Dataset (without IOWA
                and using Voting), list of the form (TN FP): ")
      (display (list (consigue-beta ln) (consigue-alpha ln)))
      (newline)
      (newline))))

;; obtain average instead of IOWA
(define damepromediogamma
  (lambda (ls)
    (cond
      ((null? ls) '())
      (else (cons (/ (+ (+ (caar ls) (car (cdr (car ls))))
                        (car (cddr (car ls)))) 3.00000000)
      (damepromediogamma (cdr ls)))))))

;; call with contp = 0
;; select >= val values
(define sinsalpos
  (lambda (lat contp val)
    (cond
      ((null? lat) contp)
      ((>= (car lat) val)
        (sinsalpos (cdr lat) (+ contp 1) val))
      (else (sinsalpos (cdr lat) contp val)))))

;; call with contp = 0
;; select < val values
(define sinsalneg
  (lambda (lat contn val)
```

```
    (cond
      ((null? lat) contn)
      ((< (car lat) val)
        (sinsalneg (cdr lat) (+ contn 1) val))
      (else (sinsalneg (cdr lat) contn val)))))


;; lista resultados de promerdios instead of IOWAs
(define manipulaprom
  (lambda (lpos lneg)
    (begin
      (newline)
      (display "Results using AVERAGE for
         pre-labelled POSs (TP FN): ")
      (display (list (sinsalpos lpos 0 0.5000)
        (sinsalneg lpos 0 0.5000)))
      (newline)
      (display "Results using AVERAGE for
         pre-labelled NEGs (TN FP): ")
      (display (list (sinsalneg lneg 0 0.5000)
        (sinsalpos lneg 0 0.5000)))
      (newline))))


;;
;;=========================================
;;=== Calc for IOWA (MODIFIED) ============
;;=========================================
;; Salidas of sor-calc-SIMPLE:
;; * resulposz: resultados de positive set. Format: ( neg pos ... neg)
;; * resulnegz: resultados de negative set. Format: ( neg pos ... pos)
;; paiowapos2: resultados POS listos para IOWA analysis. Format: (0.011 0.03  0.79),
;; ordered already from lesser to greater.
;; paiowaneg2: resultados NEG listos para IOWA analysis. Format: same as above.
;;
;; Parameter to calc-iowa-stuff is TOLERANCE. Call with Tolerance = 0.300000.
(define calc-iowa-stuff-discrete
  (lambda (tolly)
    (begin
      (set! paiowapos1 (premasajea-before-iowa (buildit4iowa-pos nbposx meposx
                                                                 listopacompap)))
      (set! paiowapos2 (extractx-vals-iowa paiowapos1))
      (set! paiowaneg1 (premasajea-before-iowa (buildit4iowa-neg nbnegx menegx
                                                                 listopacompan)))
      (set! paiowaneg2 (extractx-vals-iowa paiowaneg1))
      (set! tolerance tolly)
      (set! veci '())
      (set! iowaposs '())
      (set! supveci '())
      (set! orderednewsupvec '())
      (set! aggrega 0.0)
      (set! iowaposs (iowafunlistpos paiowapos2))
      (set! veci '())
      (set! iowanegs '())
```

```
      (set! supveci '())
      (set! orderednewsupvec '())
      (set! aggrega 0.00)
      (set! iowanegs (iowafunlistneg paiowaneg2))
      (set! negnewt (proiowaworld paiowaneg1))
      (set! posnewt (proiowaworld paiowapos1))
      (newline)
      (display "================================================================
                ============")
      (newline)
      (display "BEGINNING Results Aggregation using IOWA with Hybrid Method HACACO.")
      (newline)
      (newline)
;;      (jclassy iowanegs iowaposs)
;;      (nuevoclassy iowanegs iowaposs)
      (display "TP = ") (display (bazuka-pos iowaposs 0)) (newline)
      (display "FN = ") (display (bazuka-neg iowaposs 0)) (newline)
      (display "TN = ") (display (bazuka-neg iowanegs 0)) (newline)
      (display "FP = ") (display (bazuka-pos iowanegs 0)) (newline)
      (newline)
      (newline)
      (display "END Results Aggregation using IOWA with Hybrid Method HACACO.")
      (newline)
      (display "============================================================")
      (newline)
      (newline)
      (display "============================================================")
      (newline)
      (display "BEGINNING Results Aggregation using VOTING and AVERAGING instead
                of OWA Operators.")
      (newline)
      (dalesiniowa posnewt negnewt)
      (manipulaprom (damepromediogamma paiowapos2) (damepromediogamma paiowaneg2))
      (newline)
      (display "END Results Aggregation using VOTING and AVERAGING instead of
                OWA Operators.")
      (newline)
      (display "============================================================")
      (newline)
      (newline)
      (newline)
      )))


(define filtra-lo
  (lambda (l)
    (cond
      ((null? l) '())
      (else (cons (reverse (cdr (reverse (car l)))) (filtra-lo (cdr l)))))))


(define bazuka-pos
  (lambda (l c)
    (cond
```

```
      ((null? l) c)
      ((>= (car l) 0.5) (bazuka-pos (cdr l) (+ c 1)))
      (else (bazuka-pos (cdr l) c)))))


(define bazuka-neg
  (lambda (l c)
    (cond
      ((null? l) c)
      ((< (car l) 0.5) (bazuka-neg (cdr l) (+ c 1)))
      (else (bazuka-neg (cdr l) c)))))


  ;;=============================================
;;=== Cross-ratio Uninorm Code ================
;;=============================================
;;
;; Cross-ratio Uninorm for 2 variables
(define uxy
  (lambda (x y e)
    (cond
      ((or
         (and
           (= x 0)
           (= y 1))
         (and
           (= x 1)
           (= y 0))) 0)
      (else (/ (* (* (- 1 e) x) y)
               (+ (* (* (- 1 e) x) y) (* e (* (- 1 x) (- 1 y)))))))))

;;
;; promedio = average
(define promedio
  (lambda (x y)
    (/ (+ x y) 2.0)))


;; main call for cross-ratio uninorm test (whole dataset)
;; Parameter to calc-cross-ratio is tolly. Call with tolly = 0.500000 (identity element).
(define calc-cross-ratio
  (lambda (tolly)
    (begin
      (set! paiowapos1 (premasajea-before-iowa (buildit4iowa-pos nbposx meposx
                                                                 listopacompap)))
      (set! paiowapos2 (extractx-vals-iowa paiowapos1))
      (set! paiowaneg1 (premasajea-before-iowa (buildit4iowa-neg nbnegx menegx
                                                                 listopacompan)))
      (set! paiowaneg2 (extractx-vals-iowa paiowaneg1))
      (set! tolerance tolly)
      (newline)
;; First element only
      (display "============================================================
␣␣␣␣␣␣␣␣␣␣␣␣␣␣==============")
```

```
      (newline)
      (display "=== Cross-ratio Uninorm Vs. Average ========================
==================")
      (newline)
      (display "------------------------------------------------------------
------------------")
      (newline)
      (display "============================================================
==================")
      (newline)
      (display "============================================================
==================")
      (newline)
      (display "Element x = ") (display (car (car paiowapos2))) (newline)
      (display "Element y = ") (display (car (cdr (car paiowapos2)))) (newline)
      (newline)
      (display "Cross-ratio Uninorm = ") (display (uxy (car (car paiowapos2))
                  (car (cdr (car paiowapos2))) tolly)) (newline) (newline)
      (display "Arithmetic Mean = ") (display (promedio (car (car paiowapos2))
                                    (car (cdr (car paiowapos2))))) (newline)
      (newline)
      (display "============================================================
==================")
      )))

;;------------------------------
;;--- IOWA Calculations Code ---
;;------------------------------
;;
;;---------------------------------------
;; --- REMOVE COMMENTS FOR TESTING -------
;;---------------------------------------
;;(define thelsup '(1 1 2 3 2))
;;(define thelval '(0 0 0.3 0.7 0.9))
;;(define preveci '(0.9 0.7 0.6 0.1 0))
;;(define preveci '(1 1 1 0.5 0 0))
;;(define veci '(0 0.1 0.6 0.9 0.7))
;;(define preveci '(0.7 0.5 0.21))
;;(define tolerance 0.4)
;;(define aggrega 0.0)


;; Opposite to CONS
(define snoc
  (lambda (item lst)
    (if (null? lst) (cons item '()) (cons (car lst)
      (snoc item (cdr lst))))))

;; sort in ascending order a list of atoms
(define mylist-sort
  (lambda (lat)
    (cond
      ((null? lat) '())
      ((= (car lat) (apply min lat)) (cons (car lat)
        (mylist-sort (cdr lat))))
```

```
      (else (mylist-sort (append (cdr lat)
        (list (car lat))))))))

;; Sort in ascending order a list of pairs
;; Transform ((1 0) (1 0) (3 0.7) (2 0.9) (2 0.3)) into
;; ((1 0) (1 0) (2 0.3) (2 0.9) (3 0.7))
(define mylist-sort-pair-1st
  (lambda (lat)
    (cond
      ((null? lat) '())
      ((= (caar lat) (apply min (tranv-1st lat)))
        (cons (car lat)
        (mylist-sort-pair-1st (cdr lat))))
      (else (mylist-sort-pair-1st
        (append (cdr lat)
        (list (car lat))))))))

;; Sort in ascending order a list of pairs
;; Transform ((1 0) (1 0) (2 0.3) (3 0.7) (2 0.9)) into
;; ((1 0) (1 0) (2 0.3) (2 0.9) (3 0.7))
(define mylist-sort-pair-2nd
  (lambda (lat)
    (cond
      ((null? lat) '())
      ((= (car (cdr (car lat))) (apply min
        (tranv-2nd lat)))
        (cons (car lat)
          (mylist-sort-pair-2nd (cdr lat))))
      (else (mylist-sort-pair-2nd
        (append (cdr lat)
          (list (car lat))))))))

;; Transform ((a b) (c d) (3 y)) into (b d y)
(define tranv-2nd
  (lambda (l)
    (cond
      ((null? l) '())
      (else (cons (car (cdr (car l)))
        (tranv-2nd (cdr l))))))))

;; Transform ((a b) (c d) (3 y)) into (a c 3)
(define tranv-1st
  (lambda (l)
    (cond
      ((null? l) '())
      (else (cons (caar l)
        (tranv-1st (cdr l))))))))


;; Take lists (1 2 3) and (a b c) and create
;; ((1 a) (2 b) (3 c))
(define induced-sort-pairs
  (lambda (lsup lval)
    (cond
```

```
      ((null? lsup) '())
      (else (cons (list (car lsup) (car lval))
        (induced-sort-pairs (cdr lsup)
          (cdr lval)))))))

;; Extract the lsupport (thelsup) de las lista generada
;; por mylist-sort-pair
(define filter-thelsup
  (lambda (llat)
    (cond
      ((null? llat) '())
      (else (cons (caar llat)
        (filter-thelsup (cdr llat)))))))

;; Extract the lvalues (thelval) de las lista generada
;; por mylist-sort-pair
(define filter-thelval
  (lambda (llat)
    (cond
      ((null? llat) '())
      (else (cons (car (cdr (car llat)))
        (filter-thelval
          (cdr llat)))))))

;;-----------------------------------------
;; 'most' function evaluation Version 1
(define mostfunevalv1
  (lambda (x)
    (cond
      ((<= x 0.4) 0.0)
      ((>= x 0.9) 1.0)
      ((and (> x 0.4)
            (< x 0.9))
       (- (* x 2) 0.8))
      (else 'undefined))))

;; Evaluate the passed version of the most
;; quantifier function
(define evaluate-most
  (lambda (x myfun)
    (myfun x)))

;; Clean the output of 'evaluate-most' by reducing it
;; to 1 part integer and 2 decimals
(define clean-evaluate-most
  (lambda (value)
    (if (= (string-length (number->string value)) 3)
        (string->number (substring
          (number->string value) 0 3))
        (string->number (substring
          (number->string value) 0 4)))))

;;-------------------------------------------------------
;; 'Sup' function evaluation Version 1
```

```
(define supv1
  (lambda (x1 x2 tol)
    (if (< (abs (- x1 x2)) tol)
        1
        0)))

;; Get results (summation) for row 'k'
;; vec = list->vector lat); k = row number being worked out;
;; tol = tolerance support function;
;; cont = counter for vec length (first call, cont = 0);
;; topvec = length vector
(define addfilasupvec
  (lambda (vec k tol count topvec)
    (cond
      ((= count topvec) 0)
      ((= count (- k 1)) (addfilasupvec vec k tol
        (+ count 1) topvec))
      (else (+ (supv1 (vector-ref vec (- k 1))
        (vector-ref vec count) tol)
        (addfilasupvec vec k tol (+ count 1) topvec))))))


;; Generate Support Vector: (S1 ... Sn)
;; origlat = list of values A = (A1,..., Ak);
;; tol = tolerance
;; support function; filanum = number of row
;; (must start with 1);
;; origlat must be the original list of values A
;; (it must NOT change)
(define gen-sup-vector
  (lambda (origlat tol filanum)
    (cond
      ((> filanum (length origlat)) '())
      (else (cons (addfilasupvec
        (list->vector origlat) filanum tol 0
        (length origlat))
      (gen-sup-vector origlat tol (+ filanum 1)))))))


;; Generic function to convert LIST to VECTOR
(define conv-list-to-vec
  (lambda (lat)
    (list->vector lat)))


;;-------------------------------------------------------
;; Calcualte tis, as ti = si + 1
;; si = Support Vector 'S'
(define tis
  (lambda (si)
    (cond
      ((null? si) '())
      (else (cons (+ (car si) 1) (tis (cdr si)))))))
```

```
;; Calculate de x's that will go into the Q(x) evaluation,
;; where xi = ti / n
;; lostis = list of new tis's
;; n = length of support vector
(define x-is-tofn
  (lambda (lostis n)
    (cond
      ((null? lostis) '())
      (else (cons (inexact (/ (car lostis) n)) (x-is-tofn
                            (cdr lostis) n))))))

;; Calculate los valores de Q(x), where x = tis/ (P1)
(define qtins
  (lambda (listxs)
    (cond
      ((null? listxs) '())
      (else (cons (evaluate-most (car listxs)
        mostfunevalv1)
        (qtins (cdr listxs)))))))


;; Calculate sumatoria of all qtins (P2)
(define sumqtins
  (lambda (listqtins)
    (cond
      ((null? listqtins) 0)
      (else (+ (car listqtins) (sumqtins
        (cdr listqtins)))))))


;; Calculate P1/P2, es decir, the new support vector values
(define p1divp2
  (lambda (p1list p2)
    (cond
      ((null? p1list) '())
      (else (cons (inexact (/ (car p1list) p2)) (p1divp2
                            (cdr p1list) p2))))))


;; Generate the new Support Vector
(define newsupvec
  (lambda (milista)
    (sort < milista)))


;; Multyply 2 vectors of same length (vectors are repsented
;;  as LATs) lvec1 - Support vector list; lvec2 -
;; list of actual lvals
(define multmyvecs
  (lambda (lvec1 lvec2)
    (cond
      ((null? lvec1) 0)
      (else (+ (* (car lvec1) (car lvec2))
        (multmyvecs (cdr lvec1)
```

```
          (cdr lvec2)))))))

;;-------------------------------------------------------------
;;--- Example of generation of IOWA -------------------------
;;-------------------------------------------------------------
(define veci '())
(define supveci '())
(define orderednewsupvec '())

(define iowatest
  (lambda ()
  (begin
    (display "Vector␣original␣to␣be␣aggregated␣=␣")
    (write preveci) (newline)
    (set! supveci (gen-sup-vector preveci tolerance 1))
    (display "Support␣Vector␣=␣")
    (write supveci) (newline)
    (display "With␣a␣Tolerance␣of:␣")
    (write tolerance) (newline)
    (set! veci (filter-thelval
      (mylist-sort-pair-1st (mylist-sort-pair-2nd
      (induced-sort-pairs supveci preveci)))))
    (display "The␣Induced␣Similarity␣Order␣Vector
␣␣␣␣␣␣being␣aggregated␣=␣")
    (write veci) (newline)
    (set! orderednewsupvec
      (newsupvec
        (p1divp2 (qtins (x-is-tofn (tis supveci)
          (length (tis supveci))))
        (sumqtins (qtins (x-is-tofn (tis supveci)
          (length (tis supveci))))))))
    (display "New␣Support␣Vector␣=␣")
    (write orderednewsupvec)
    (newline)
    (display "Aggregation␣=␣")
    (set! aggrega (multmyvecs orderednewsupvec veci))
    (write aggrega)
    (newline)
    (newline))))
```

C.5 Support Code

```
;; are all members of a list vecs?
(define allvecs?
  (lambda (l)
    (cond
      ((null? l) #t)
      (else
        (if (vector? (car l))
            (allvecs? (cdr l))
            #f)))))

;; are all members of a list pairs?
```

```
(define allpair?
  (lambda (l)
    (cond
      ((null? l) #t)
      (else
        (if (pair? (car l))
            (allpair? (cdr l))
            #f)))))

;; is value >= 1
(define isvalue>=1
  (lambda (lista)
    (cond
      ((null? lista) '())
      (else (if (>= (car (cdr (car lista))) 1.00)
                (cons (car lista)
                 (isvalue>=1 (cdr lista)))
                (isvalue>=1 (cdr lista)))))))

;; is value <= 0
(define isvalue<0
  (lambda (lista)
    (cond
      ((null? lista) '())
      (else (if (< (car (cdr (car lista))) 0.00)
                (cons (car lista)
                 (isvalue<0 (cdr lista)))
                (isvalue<0 (cdr lista)))))))

;; is value = 0
(define isvalue=0
  (lambda (lista)
    (cond
      ((null? lista) '())
      (else (if (= (car (cdr (car lista))) 0.00)
                (cons (car lista)
                 (isvalue=0 (cdr lista)))
                (isvalue=0 (cdr lista)))))))

;; resize list of vectors mixed (5 and 6 length)
;;   to all length = 8 vectors
(define resizevector5to8
  (lambda (l)
    (cond
      ((null? l) '())
      ((= (vector-length (car l)) 8)
        (cons (car l) (resizevector5to6 (cdr l))))
      (else (cons (inspectionvec (car l))
        (resizevector5to8 (cdr l)))))))

;;Re-write a vector of length 5, extending it
;; to lenght 6, adding
;; symbol 'noinfo in
;; vec-ref = 2, 3, 4, 5, 6 and 7
```

```
(define inspectionvec
  (lambda (vec)
    (if (= (vector-ref vec 2) -1)
        (list->vector (list (vector-ref vec 0)
                            (vector-ref vec 1)
                            'nopsc
                            'nonsc
                            'nocobj
                            'nocsor
                            'nomaxdist
                            'nomindist
                            ))
        (list->vector (list (vector-ref vec 0)
                            (vector-ref vec 1)
                            (vector-ref vec 2)
                            (vector-ref vec 3)
                            (vector-ref vec 4)
                            'nocsor
                            'nomaxdist
                            'nomindist
                            )))))

;; check that all vectors in the list are of lenghth = 8
(define alllength8?
  (lambda (l)
    (cond
      ((null? l) #t)
      (else (if (= (vector-length (car l)) 8)
                #t
                #f)))))

;; are both list the same with respect to the first element
;; of the vector
;; the words in the list of vectors
(define samelovecs?
  (lambda (lv1 lv2)
    (if (= (length lv1) (length lv2))
        (samelovecsaux lv1 lv2)
        #f)))

;; aux function of samelovecs? compares heads of
;; both vectors
(define samelovecsaux
  (lambda (lv1 lv2)
    (cond
      ((null? lv1) #t)
      ((not (equal? (vector-ref
        (car lv1) 0) (vector-ref (car lv2) 0))) #f)
      (else (samelovecsaux (cdr lv1) (cdr lv2))))))

;; valida si hay algun elemento del vector con 'x' en
;;   lugar de a valid PoS
(define isthere-a-nonpos?
  (lambda (lvec)
```

```
    (cond
      ((null? lvec) #f)
      ((equal? (vector-ref (car lvec) 1) 'x) #t)
      (else  (isthere-a-nonpos? (cdr lvec))))))


;; replace the x in vector-ref 1: PoS tag
(define replacepost
  (lambda (lvec)
    (cond
      ((null? lvec) '())
      (else (if (equal? (vector-ref (car lvec) 1) 'x)
                (cons (dorep (car lvec))
                  (replacepost (cdr lvec)))
                (cons (car lvec)
                  (replacepost (cdr lvec))))))))

;;
(define dorep
  (lambda (vec)
    (list->vector (list (vector-ref vec 0)
                        'nopostag
                        (vector-ref vec 2)
                        (vector-ref vec 3)
                        (vector-ref vec 4)
                        (vector-ref vec 5)
                        (vector-ref vec 6)
                        (vector-ref vec 7)))))


; resize list of vectors of length 8) to all
;;   length = 9 vectors
(define resizevector8to9
  (lambda (l)
    (cond
      ((null? l) '())
      (else (cons (inspectionvec9 (car l))
        (resizevector8to9 (cdr l)))))))

;;Re-write a vector of length 8, extending it to
;; lenght 9, adding
;; value 0 for learning field vector-ref is 8
(define inspectionvec9
  (lambda (vec)
    (list->vector (list (vector-ref vec 0)
                        (vector-ref vec 1)
                        (vector-ref vec 2)
                        (vector-ref vec 3)
                        (vector-ref vec 4)
                        (vector-ref vec 5)
                        (vector-ref vec 6)
                        (vector-ref vec 7)
                        0))))
```

```
;; check that all vectors in the list are of lenghth = 9
(define alllength9?
  (lambda (l)
    (cond
      ((null? l) #t)
      (else (if (= (vector-length (car l)) 9)
                #t
                #f)))))


;; Add semantic orientation to lexicon
;; (lex = lexicon; lso = list of SO)
(define addso
  (lambda (lso lex)
    (if (null? lso)
        '()
        (cons (performaddso
          (car lso) lex) (addso (cdr lso) lex)))))

;; Perform addso
(define performaddso
  (lambda (lst lex)
    (cond
      ((null? lex) '())
      ((equal? (car lst) (vector-ref (car lex) 0))
        (cons (constructvec (car (cdr lst))
          (car lex)) (performaddso lst (cdr lex))))
      (else (performaddso lst (cdr lex))))))

;; Constructvec with SO added
(define constructvec
  (lambda (valor vec)
    (list->vector (list (vector-ref vec 0)
                        (vector-ref vec 1)
                        (vector-ref vec 2)
                        (vector-ref vec 3)
                        (vector-ref vec 4)
                        valor
                        (vector-ref vec 6)
                        (vector-ref vec 7)
                        (vector-ref vec 8)))))

;; intersection vacia de two lvecs (lexicons)
(define emptyintersect?
  (lambda (lvec1 lvec2)
    (cond
      ((null? lvec1) #t)
      ((estaenlvec2? (car lvec1) lvec2) #f)
      (else (emptyintersect? (cdr lvec1) lvec2)))))

;; Auxiliar for emptyintersect?
(define estaenlvec2?
  (lambda (vec lvec)
    (cond
```

```
      ((null? lvec) #f)
      ((equal? (vector-ref vec 0)
        (vector-ref (car lvec) 0))
      (begin (display vec) (newline) #t))
      (else (estaenlvec2? vec (cdr lvec))))))

;;-------------------------------------------------------
;; Programs utilities para actualizar y retrieve
;;   partes del vec de lexicon
;;-------------------------------------------------------
(define getword
  (lambda (vec)
    (vector-ref vec 0)))

(define getpos
  (lambda (vec)
    (vector-ref vec 1)))

(define getposscore
  (lambda (vec)
    (vector-ref vec 2)))

(define getnegscore
  (lambda (vec)
    (vector-ref vec 3)))

(define getobjscore
  (lambda (vec)
    (vector-ref vec 4)))

(define getsor
  (lambda (vec)
    (vector-ref vec 5)))

(define getmaxdist
  (lambda (vec)
    (vector-ref vec 6)))

(define getmindist
  (lambda (vec)
    (vector-ref vec 7)))

(define getupdtindex
  (lambda (vec)
    (vector-ref vec 8)))

;;-------------------------------------------------------------
;; Programs de match y reemplazo de SOs, Min and Max
;;-------------------------------------------------------------
;; Replace MAXDIST end lvecs: retorna la lista de vectores,
;; para aquellos vectores que satisfacen la comparacion
(define replacemaxdist
  (lambda (lmax lvec)
    (cond
```

```
      ((null? lmax) '())
      ((foundit? (car lmax) lvec)
        (cons (fixvector6
          (deliverit (car lmax) lvec)
          (car (cdr (car lmax))))
        (replacemaxdist (cdr lmax) lvec)))
      (else (replacemaxdist (cdr lmax) lvec)))))

;; Replace MINDIST end lvecs: retorna la lista de vectores,
;; para aquellos vectores que satisfacen la comparacion
(define replacemindist
  (lambda (lmax lvec)
    (cond
      ((null? lmax) '())
      ((foundit? (car lmax) lvec)
        (cons (fixvector7
          (deliverit (car lmax) lvec)
          (car (cdr (car lmax))))
        (replacemindist (cdr lmax) lvec)))
      (else (replacemindist (cdr lmax) lvec)))))

;; Replace CSOR end lvecs: retorna la lista de vectores,
;; para aquellos vectores que satisfacen la comparacion
(define replacecsor
  (lambda (lmax lvec)
    (cond
      ((null? lmax) '())
      ((foundit? (car lmax) lvec)
        (cons (fixvector5
          (deliverit (car lmax) lvec)
          (car (cdr (car lmax))))
        (replacecsor (cdr lmax) lvec)))
      (else (replacecsor (cdr lmax) lvec)))))

;; check and returns #t si el vector buscado pertence a lvecs
(define foundit?
  (lambda (lst lvec)
    (cond
      ((null? lvec) #f)
      ((equal? (car lst) (getword (car lvec))) #t)
      (else (foundit? lst (cdr lvec))))))

;; si foundit? is #t, retorna el "vec"  sobre el
;; cual se debe hacer el reemplazo de valor
(define deliverit
  (lambda (lst lvec)
    (cond
      ((null? lvec) '())
      ((equal? (car lst) (getword (car lvec))) (car lvec))
      (else (deliverit lst (cdr lvec))))))

;; remove los '() que haya en una lista de vectores
(define removeemptysets
  (lambda (lvecs)
```

```
    (if (equal? (car lvecs) '())
        (removeemptysets (cdr lvecs))
        (cons (car lvecs) (removeemptysets (cdr lvecs))))))

;; devuelve el vector con el valor deseado ya reemplazado
;;   para vec index 5: CSOR
(define fixvector5
  (lambda (vec valor)
    (list->vector (list (vector-ref vec 0)
                        (vector-ref vec 1)
                        (vector-ref vec 2)
                        (vector-ref vec 3)
                        (vector-ref vec 4)
                        valor
                        (vector-ref vec 6)
                        (vector-ref vec 7)
                        (vector-ref vec 8)))))

;; devuelve el vector con el valor deseado ya reemplazado
;;   para vec index 6: MAXDIST
(define fixvector6
  (lambda (vec valor)
    (list->vector (list (vector-ref vec 0)
                        (vector-ref vec 1)
                        (vector-ref vec 2)
                        (vector-ref vec 3)
                        (vector-ref vec 4)
                        (vector-ref vec 5)
                        valor
                        (vector-ref vec 7)
                        (vector-ref vec 8)))))

;; devuelve el vector con el valor deseado ya reemplazado
;;   para vec index 7: MINDIST
(define fixvector7
  (lambda (vec valor)
    (list->vector (list (vector-ref vec 0)
                        (vector-ref vec 1)
                        (vector-ref vec 2)
                        (vector-ref vec 3)
                        (vector-ref vec 4)
                        (vector-ref vec 5)
                        (vector-ref vec 6)
                        valor
                        (vector-ref vec 8)))))

;; reemplaza en la lista de vectores del lexicon,
;; los vectores provistos en la lista de vectores:
;;   lvecs: INCOMPLETO
(define updateellexicon
  (lambda (lvecs lex)
    (cond
      ((null? lvecs) '())
      ((estaen? (car lvecs) lex)
```

```
        (cons (fixlex (car lvecs) lex)
        (updateellexicon (cdr lvecs) lex)))
      (else (updateellexicon (cdr lvecs) lex)))))


(define estaen?
  (lambda (vec lex)
    (cond
      ((null? lex) #f)
      ((equal? (getword vec) (getword (car lex))) #t)
      (else (estaen? vec (cdr lex))))))

(define fixlex
  (lambda (vec lex)
    (cond
      ((null? lex) '())
      ((equal? (getword vec) (getword (car lex))) vec)
      (else (fixlex vec (cdr lex))))))


(define loopcorrections
  (lambda (l1 l2)
    (cond
      ((null? l1) '())
      (else (loopcorrections (cdr l1)
        (looplex (car l1) l2))))))

(define looplex
  (lambda (vec l)
    (cond
      ((null? l) '())
      ((equal? (getword vec) (getword (car l)))
        (cons vec (looplex vec (cdr l))))
      (else (cons (car l)
        (looplex vec (cdr l)))))))

(define use-sub-sor
  (lambda (new old lvec)
    (sub-sor (car (cdr  (car losso)))
      (car (car losso)) losnegs)))

;;-----------------------------------------------------------------
;;-----------------------------------------------------------------
;; llamar a cambios en neglex con SOR
(define cambios-j-losnegs-sor
  (lambda (lst)
    (cond
      ((null? lst) 'done)
      (else (begin
              (set! losnegs (sub-sor (car lst) losnegs))
              (cambios-j-losnegs-sor (cdr lst)))))))

; llamar a cambios en neglex con MAX
(define cambios-j-losnegs-max
```

```
  (lambda (lst)
    (cond
      ((null? lst) 'done)
      (else (begin
              (set! losnegs (sub-max (car lst) losnegs))
              (cambios-j-losnegs-max (cdr lst)))))))

; llamar a cambios en neglex con MIN
(define cambios-j-losnegs-min
  (lambda (lst)
    (cond
      ((null? lst) 'done)
      (else (begin
              (set! losnegs (sub-min (car lst) losnegs))
              (cambios-j-losnegs-min (cdr lst)))))))

;; llamar a cambios en poslex con SOR
(define cambios-j-losposs-sor
  (lambda (lst)
    (cond
      ((null? lst) 'done)
      (else (begin
              (set! losposs (sub-sor (car lst) losposs))
              (cambios-j-losposs-sor (cdr lst)))))))

; llamar a cambios en poslex con MAX
(define cambios-j-losposs-max
  (lambda (lst)
    (cond
      ((null? lst) 'done)
      (else (begin
              (set! losposs (sub-max (car lst) losposs))
              (cambios-j-losposs-max (cdr lst)))))))

; llamar a cambios en poslex con MIN
(define cambios-j-losposs-min
  (lambda (lst)
    (cond
      ((null? lst) 'done)
      (else (begin
              (set! losposs (sub-min (car lst) losposs))
              (cambios-j-losposs-min (cdr lst)))))))


;; recibe lista de cambio puntual, como '(poor valor) y
;;   un lexicon (lista de vectores)
(define sub-sor
  (lambda (lstv lvec)
    (cond
      ((null? lvec) '())
      ((equal? (getword (car lvec)) (car lstv))
        (cons (fixvectorcito5
          (car lvec) (car (cdr lstv)))
        (sub-sor lstv (cdr lvec))))
```

```
      (else (cons (car lvec)
        (sub-sor lstv (cdr lvec)))))))

;; recibe lista de cambio puntual, como '(poor valor)
;;   y un lexicon (lista de vectores)
(define sub-max
  (lambda (lstv lvec)
    (cond
      ((null? lvec) '())
      ((equal? (getword (car lvec)) (car lstv))
        (cons (fixvectorcito6 (car lvec) (car (cdr lstv)))
          (sub-max lstv (cdr lvec))))
      (else (cons (car lvec) (sub-max lstv (cdr lvec)))))))

; recibe lista de cambio puntual, como '(poor valor) y un
;;   lexicon (lista de vectores)
(define sub-min
  (lambda (lstv lvec)
    (cond
      ((null? lvec) '())
      ((equal? (getword (car lvec)) (car lstv))
        (cons (fixvectorcito7
          (car lvec) (car (cdr lstv)))
        (sub-min lstv (cdr lvec))))
      (else (cons (car lvec)
        (sub-min lstv (cdr lvec)))))))


(define fixvectorcito5
  (lambda (vec val)
    (list->vector (list (vector-ref vec 0)
                        (vector-ref vec 1)
                        (vector-ref vec 2)
                        (vector-ref vec 3)
                        (vector-ref vec 4)
                        val
                        (vector-ref vec 6)
                        (vector-ref vec 7)
                        (vector-ref vec 8)))))

(define fixvectorcito6
  (lambda (vec val)
    (list->vector (list (vector-ref vec 0)
                        (vector-ref vec 1)
                        (vector-ref vec 2)
                        (vector-ref vec 3)
                        (vector-ref vec 4)
                        (vector-ref vec 5)
                        val
                        (vector-ref vec 7)
                        (vector-ref vec 8)))))

(define fixvectorcito7
  (lambda (vec val)
```

```
    (list->vector (list (vector-ref vec 0)
                        (vector-ref vec 1)
                        (vector-ref vec 2)
                        (vector-ref vec 3)
                        (vector-ref vec 4)
                        (vector-ref vec 5)
                        (vector-ref vec 6)
                        val
                        (vector-ref vec 8)))))

;; si se desea cambiar adjetivo satellite de 's' a 'a'
(define fix-satellite-vec
  (lambda (vec)
    (list->vector (list (vector-ref vec 0)
                        'a
                        (vector-ref vec 2)
                        (vector-ref vec 3)
                        (vector-ref vec 4)
                        (vector-ref vec 5)
                        (vector-ref vec 6)
                        (vector-ref vec 7)
                        (vector-ref vec 8)
                        (vector-ref vec 9)))))

;; rutina que hace el loop (si se quiere reemplazar
;;      adjetivos satelitales)
(define fix-satellite-los
  (lambda (lst)
    (cond
      ((null? lst) '())
      ((equal? (getpos (car lst)) 's) (cons
        (fix-satellite-vec (car lst))
        (fix-satellite-los (cdr lst))))
      (else (cons (car lst)
        (fix-satellite-los (cdr lst)))))))


;;-------------------------------------------------------------------------
;;-------------------------------------------------------------------------
;; STATISTICS
(define statsmine
  (lambda (lst)
    (cond
      ((null? lst) 0)
      ((number? (getupdtindex (car lst)))
        (+ 1 (statsmine (cdr lst))))
      (else (statsmine (cdr lst))))))

(define statspos
  (lambda (lst)
    (cond
      ((null? lst) 0)
      ((or
           (equal? (getpos (car lst)) 'n)
```

```
            (equal? (getpos (car lst)) 'a)
            (equal? (getpos (car lst)) 's)
            (equal? (getpos (car lst)) 'v)
            (equal? (getpos (car lst)) 'r))
       (+ 1 (statspos (cdr lst))))
       (else (statspos (cdr lst))))))

(define itemswmax
  (lambda (lvecs)
    (cond
      ((null? lvecs) '())
      ((number? (getmaxdist (car lvecs)))
        (cons (car lvecs) (itemswmax (cdr lvecs))))
      (else (itemswmax (cdr lvecs))))))

(define itemswmin
  (lambda (lvecs)
    (cond
      ((null? lvecs) '())
      ((number? (getmindist (car lvecs)))
        (cons (car lvecs) (itemswmin (cdr lvecs))))
      (else (itemswmin (cdr lvecs))))))

(define itemswnoposscore
  (lambda (lvecs)
    (cond
      ((null? lvecs) '())
      ((number? (getposscore (car lvecs)))
        (itemswnoposscore (cdr lvecs)))
      (else (cons (car lvecs)
        (itemswnoposscore (cdr lvecs)))))))

;; call with dmax = -1
(define themax
  (lambda (l dmax)
    (cond
      ((null? l) dmax)
      ((not (number? (getposscore (car l))))
        (themax (cdr l) dmax))
      ((> (getposscore (car l)) dmax) (themax (cdr l)
        (getposscore (car l))))
      (else (themax (cdr l) dmax)))))



;;-----------------------------------------------------------------------
;;--- Routines for processing the results of Supervised
;;    Learning techniques ---
;;-----------------------------------------------------------------------
(define wrong-positives-count
  (lambda (l)
    (cond
      ((null? l) 0)
      ((and (equal? (car (cdr (cdr (car l)))) 'neg)
            (eq? (car (cdr (cdr (cdr (car l))))) 1))
```

```
            (+ 1 (wrong-positives-count (cdr l))))
      (else (wrong-positives-count (cdr l))))))

(define true-positives-count
  (lambda (l)
    (cond
      ((null? l) 0)
      ((and (equal? (car (cdr (cdr (car l)))) 'pos)
            (eq? (car (cdr (cdr (cdr (car l))))) 1))
            (+ 1 (true-positives-count (cdr l))))
      (else (true-positives-count (cdr l))))))

(define wrong-negatives-count
  (lambda (l)
    (cond
      ((null? l) 0)
      ((and (equal? (car (cdr (cdr (car l)))) 'pos)
            (eq? (car (cdr (cdr (cdr (car l))))) 0))
            (+ 1 (wrong-negatives-count (cdr l))))
      (else (wrong-negatives-count (cdr l))))))

(define true-negatives-count
  (lambda (l)
    (cond
      ((null? l) 0)
      ((and (equal? (car (cdr (cdr (car l)))) 'neg)
            (eq? (car (cdr (cdr (cdr (car l))))) 0))
            (+ 1 (true-negatives-count (cdr l))))
      (else (true-negatives-count (cdr l))))))

(define wrong-positives-extract
  (lambda (l)
    (cond
      ((null? l) '())
      ((and (equal? (car (cdr (cdr (car l)))) 'neg)
            (eq? (car (cdr (cdr (cdr (car l))))) 1))
      (cons (car l) (wrong-positives-extract (cdr l))))
      (else (wrong-positives-extract (cdr l))))))

(define true-positives-extract
  (lambda (l)
    (cond
      ((null? l) '())
      ((and (equal? (car (cdr (cdr (car l)))) 'pos)
            (eq? (car (cdr (cdr (cdr (car l))))) 1))
      (cons (car l) (true-positives-extract (cdr l))))
      (else (true-positives-extract (cdr l))))))

(define wrong-negatives-extract
  (lambda (l)
    (cond
      ((null? l) '())
      ((and (equal? (car (cdr (cdr (car l)))) 'pos)
            (eq? (car (cdr (cdr (cdr (car l))))) 0))
```

```
      (cons (car l) (wrong-negatives-extract (cdr l))))
      (else (wrong-negatives-extract (cdr l))))))

(define true-negatives-extract
  (lambda (l)
    (cond
      ((null? l) '())
      ((and (equal? (car (cdr (cdr (car l)))) 'neg)
            (eq? (car (cdr (cdr (cdr (car l))))) 0))
      (cons (car l) (true-negatives-extract (cdr l))))
      (else (true-negatives-extract (cdr l))))))

(define cuantos-orignegs
  (lambda (l)
    (cond
      ((null? l) 0)
      ((eq? (car (cdr (cdr (cdr (car l))))) 0)
        (+ 1 (cuantos-orignegs (cdr l))))
      (else (cuantos-orignegs (cdr l))))))

(define cuantos-origposs
  (lambda (l)
    (cond
      ((null? l) 0)
      ((eq? (car (cdr (cdr (cdr (car l))))) 1)
       (+ 1 (cuantos-origposs (cdr l))))
      (else (cuantos-origposs (cdr l))))))


(define siemprenum
  (lambda (l)
    (cond
      ((null? l) #t)
      ((number? (car (car l)))
        (siemprenum (cdr l)))
      (else #f))))

(define siemprestring
  (lambda (l)
    (cond
      ((null? l) #t)
      ((string? (car (cdr (car l))))
        (siemprestring (cdr l)))
      (else #f))))

;;-----------------------------------
;;--- Validation lexicons -----------
;;-----------------------------------
(define repetidas
  (lambda (l1 l2)
    (cond
      ((null? l1) '())
      (else
        (cons (estarep (getword (car l1)) l2)
```

```
        (repetidas (cdr l1) l2))))))

(define estarep
  (lambda (wrd lst)
    (cond
      ((null? lst) '())
      ((equal? (getword (car lst)) wrd)
        (car lst))
      (else (estarep wrd (cdr lst))))))

(define quitavacios
  (lambda (lavec)
    (cond
      ((null? lavec) '())
      ((and
         (not (vector? (car lavec)))
         (equal? (car lavec) '()))
      (quitavacios
         (cdr lavec)))
      (else (cons (car lavec)
        (quitavacios (cdr lavec)))))))

;;
;;
;;============================ MEGA Global Variables ===========
;; "thelex" is The Opinion Lexicon
;; "neglex" is the NEG Opinion Lexicon
;; "poslex" is the POS Opinion Lexicon
;; "objlex" is the OBJ Opinion Lexicon
;;============================================================
;;
;;------------------------------------------------------------
;; limpiador: aplica all functions to process rules
;;------------------------------------------------------------
(define limpiador-x
  (lambda (lst)
    (manejador-x (quita-basura (find-negation (dealwith-unless
      (dealwith-despite
       (dealwith-but (extract-essentials lst)))))))))

;;------------------------------------------------------------
;;
;;------------------------------------------------------------
;; JOIN LEXICONS
;;------------------------------------------------------------
;; join pos and neg lexicons
(define join-lexs
  (lambda (l1 l2)
    (append l1 l2)))

;; add label to lexicons and create per occurrence (v label)
(define add-label
  (lambda (label l)
    (cond
      ((null? l) '())
```

```
      (else (cons (list (car l) label)
        (add-label label (cdr l)))))))

;;test predicate for new lexicon
(define testnewlex?
  (lambda (l)
    (cond
      ((null? l) #t)
      ((and
         (vector? (car (car l)))
         (or
           (equal? (car (cdr (car l))) 'neg)
           (equal? (car (cdr (car l))) 'pos)))
      (testnewlex? (cdr l)))
      (else #f))))

;;-------------------------------------------------------------
;;
;; take a list of the form (a b) ands replace "b"
;; by the right tag in PoS
;;
;; (n=nouns, v=verbs, a=adjectives, r=adverbs)
;; (CC) (conjunction AND, OR, BUT) (c)
;; (EX) (existential THERE) (x)
;; (MD) (modal CAN, SHOULD, WILL) (m)
;; (ABL) (pre-qualifier QUITE, RATHER)(q)
;; (QL) (qualifier VERY FAIRLY) (l)
;; (CS) (conjunction IF, ALTHOUGH) (i)
;; (IN) (prepositions, DESPITE) (p)
;;
(define extract-essentials
  (lambda (lst)
     (removenullsall (dodofiltergood
      (dodoreplacegood lst)))))

;; replace the proper tags
(define replacegood
  (lambda (l)
    (cond
      ((or
         (equal? (car (cdr l)) 'BE)
         (equal? (car (cdr l)) 'BED)
         (equal? (car (cdr l)) 'BED*)
         (equal? (car (cdr l)) 'BEDZ)
         (equal? (car (cdr l)) 'BEDZ*)
         (equal? (car (cdr l)) 'BEG)
         (equal? (car (cdr l)) 'BEM)
         (equal? (car (cdr l)) 'BEM*)
         (equal? (car (cdr l)) 'BEN)
         (equal? (car (cdr l)) 'BER)
         (equal? (car (cdr l)) 'BER*)
         (equal? (car (cdr l)) 'BEZ)
         (equal? (car (cdr l)) 'BEZ*)
         (equal? (car (cdr l)) 'DO)
```

```
    (equal? (car (cdr l)) 'DO*)
    (equal? (car (cdr l)) 'DOD)
    (equal? (car (cdr l)) 'DOD*)
    (equal? (car (cdr l)) 'DOZ)
    (equal? (car (cdr l)) 'DDZ) ;; weird acronym
    (equal? (car (cdr l)) 'DOZ*)
    (equal? (car (cdr l)) 'HV)
    (equal? (car (cdr l)) 'HV*)
    (equal? (car (cdr l)) 'HVD)
    (equal? (car (cdr l)) 'HVD*)
    (equal? (car (cdr l)) 'HVG)
    (equal? (car (cdr l)) 'HVN)
    (equal? (car (cdr l)) 'HVZ)
    (equal? (car (cdr l)) 'HVZ*)
    (equal? (car (cdr l)) 'VB)
    (equal? (car (cdr l)) 'VB-HL)
    (equal? (car (cdr l)) 'VBD)
    (equal? (car (cdr l)) 'VBG)
    (equal? (car (cdr l)) 'VBN)
    (equal? (car (cdr l)) 'VBN-HL)
    (equal? (car (cdr l)) 'VBZ-HL)
    (equal? (car (cdr l)) 'VHB) ;; weird acronym
    (equal? (car (cdr l)) 'VBZ))
            (list (car l) 'v))
((or
    (equal? (car (cdr l)) 'NN)
    (equal? (car (cdr l)) 'NN$)
    (equal? (car (cdr l)) 'NNS)
    (equal? (car (cdr l)) 'NNS-HL)
    (equal? (car (cdr l)) 'NN-HL)
    (equal? (car (cdr l)) 'NNS$)
    (equal? (car (cdr l)) 'NP)
    (equal? (car (cdr l)) 'NP$)
    (equal? (car (cdr l)) 'NPS)
    (equal? (car (cdr l)) 'NPS$)
    (equal? (car (cdr l)) 'NR)
    (equal? (car (cdr l)) 'NR$)
    (equal? (car (cdr l)) 'NRS))
            (list (car l) 'n))
((or
    (equal? (car (cdr l)) 'RB)
    (equal? (car (cdr l)) 'EX)
    (equal? (car (cdr l)) 'RB$)
    (equal? (car (cdr l)) 'RBR)
    (equal? (car (cdr l)) 'RBT)
    (equal? (car (cdr l)) 'RN)
    (equal? (car (cdr l)) 'dtx)
    (equal? (car (cdr l)) 'abl)
    (equal? (car (cdr l)) 'ap)
    (equal? (car (cdr l)) 'RP))
            (list (car l) 'r))
((or
    (equal? (car (cdr l)) 'JJ)
    (equal? (car (cdr l)) 'jj-hl)
```

```
          (equal? (car (cdr l)) 'JJ$)
          (equal? (car (cdr l)) 'JJR)
          (equal? (car (cdr l)) 'JJS)
          (equal? (car (cdr l)) 'JJT))
                  (list (car l) 'a))
       ((equal? (car (cdr l)) 'CC)
                  (list (car l) 'c))
       ((equal? (car (cdr l)) 'CC-HL)
                  (list (car l) 'i))
       ((equal? (car (cdr l)) 'EX)
                  (list (car l) 'x))
       ((equal? (car (cdr l)) 'MD)
                  (list (car l) 'm))
        ((equal? (car (cdr l)) 'abx)
                  (list (car l) 'b))
       ((equal? (car (cdr l)) 'QL)
                  (list (car l) 'l))
       ((equal? (car (cdr l)) 'QLP)
                  (list (car l) 'l))
       ((equal? (car (cdr l)) 'CS)
                  (list (car l) 'i))
       ((equal? (car (cdr l)) 'IN)
                  (list (car l) 'p))
       (else l))))

;; do replacegood for all the listof (a b)'s
(define doreplacegood
  (lambda (l)
    (cond
      ((null? l) '())
      (else (cons (replacegood (car l))
        (doreplacegood (cdr l)))))))

;; call doreplacegood for the whole list of
;;  ((a b) (x y)) ... ((a1 b1) (x y))
(define dodoreplacegood
  (lambda (l)
    (cond
      ((null? l) '())
      (else (cons (doreplacegood (car l))
        (dodoreplacegood (cdr l)))))))

;;
(define filternoness
  (lambda (l)
    (cond
      ((or
         (equal? (car (cdr l)) 'n)
         (equal? (car (cdr l)) 'v)
         (equal? (car (cdr l)) 'a)
         (equal? (car (cdr l)) 'r)
         (equal? (car (cdr l)) 'ninv)
         (equal? (car (cdr l)) 'vinv)
         (equal? (car (cdr l)) 'ainv)
```

```
          (equal? (car (cdr l)) 'rinv)
          (equal? (car (cdr l)) 'c)
          (equal? (car (cdr l)) 'x)
          (equal? (car (cdr l)) 'm)
          (equal? (car (cdr l)) 'q)
          (equal? (car (cdr l)) 'l)
          (equal? (car (cdr l)) 'i)
          (equal? (car (cdr l)) 'p)
          (equal? (car (cdr l)) 'negation)
          (equal? (car (cdr l)) 'punto)) l)
      (else '())))

;;
(define dofiltergood
  (lambda (l)
    (cond
      ((null? l) '())
      (else (cons (filternoness (car l))
        (dofiltergood
        (cdr l)))))))

(define dodofiltergood
  (lambda (l)
    (cond
      ((null? l) '())
      (else (cons (dofiltergood (car l))
        (dodofiltergood
        (cdr l)))))))

;;
(define removenulls
  (lambda (l)
    (cond
      ((null? l) '())
      ((equal? (car l) '())
        (removenulls (cdr l)))
      (else (cons (car l)
        (removenulls (cdr l)))))))

(define removenullsall
  (lambda (l)
    (cond
      ((null? l) '())
      (else (cons (removenulls (car l))
        (removenullsall
        (cdr l)))))))

;;
(define isthere-negation?
  (lambda (lst)
    (cond
      ((null? lst) #f)
      ((equal? (car (cdr (car lst)))
        'negation) #t)
```

```
      (else (isthere-negation?
        (cdr lst))))))
;;
(define find-negation
  (lambda (lst)
    (cond
      ((null? lst) '())
      ((isthere-negation? (car lst))
        (cons (clean-negation
        (car lst)) (find-negation (cdr lst))))
      (else (cons (car lst) (find-negation
        (cdr lst)))))))

;; remove the 'negation' particle
(define clean-negation
  (lambda (lst)
     (let [(pasada1 (append (negation-b lst 0)
      (clean-negation-aux
      (negation-a lst 0) 0)))]
       (let [(pasada2 (append (negation-b pasada1 0)
        (clean-negation-aux
          (negation-a pasada1 0) 0)))]
         (append (negation-b pasada2 0)
          (clean-negation-aux
            (negation-a pasada2 0) 0))))))

;; get sub-sentence BEFORE the 'negation particle.
;; Call first with sem=0
(define negation-b
  (lambda (l sem)
    (cond
      ((or
         (null? l)
         (= sem 1)) '())
      ((equal? (car (cdr (car l))) 'negation)
        (negation-b l 1))
      (else (cons (car l)
        (negation-b (cdr l) sem))))))

;; get sub-sentence AFTER the 'negation particle
;; (excluding it). Call first with sem=0
(define negation-a
  (lambda (l sem)
    (cond
      ((null? l) '())
      ((and
         (= sem 0)
         (equal? (car (cdr (car l))) 'negation))
      (negation-a (cdr l) 1))
      ((= sem 1) (cons (car l)
        (negation-a (cdr l) sem)))
      (else (negation-a (cdr l) sem)))))

;; this is the routine that process the sub-sentence after
```

```
;; the negation particle and generate the inversion particles in the sentence
(define clean-negation-aux
  (lambda (lst sem)
    (cond
      ((null? lst) '())
      ((and
         (= sem 0)
         (equal? (car (cdr (car lst))) 'n))
      (cons (list (caar lst) 'ninv)
         (clean-negation-aux (cdr lst) 1)))
      ((and
         (= sem 0)
         (equal? (car (cdr (car lst))) 'v))
      (cons (list (caar lst) 'vinv)
         (clean-negation-aux (cdr lst) 1)))
      ((and
         (= sem 0)
         (equal? (car (cdr (car lst))) 'a))
      (cons (list (caar lst) 'ainv)
         (clean-negation-aux (cdr lst) 1)))
      ((and
         (= sem 0)
         (equal? (car (cdr (car lst))) 'r))
      (cons (list (caar lst) 'rinv)
         (clean-negation-aux (cdr lst) 1)))
      (else (cons (car lst)
        (clean-negation-aux (cdr lst) sem))))))

;; look for inverted polarity words in a sentence.
;; Returns #t if finds an inverted particle, otherwise returns #f
(define find-inverted-x?
  (lambda (l)
    (cond
      ((null? l) #f)
      ((or
         (equal? (car (cdr (car l))) 'ninv)
         (equal? (car (cdr (car l))) 'vinv)
         (equal? (car (cdr (car l))) 'ainv)
         (equal? (car (cdr (car l))) 'rinv)) #t)
      (else (find-inverted-x? (cdr l))))))

;; returns sentences with an inverted particles present *only*
(define write-inverteds
  (lambda (lst)
    (cond
      ((null? lst) '())
      ((find-inverted-x? (car lst)) (cons (car lst)
        (write-inverteds (cdr lst))))
      (else (write-inverteds (cdr lst))))))


;; pick only relevant pairs from a sentence
(define solo-relevant-sent
  (lambda (l)
```

```
    (cond
      ((null? l) '())
      ((or
         (equal? (car (cdr (car l))) 'ninv)
         (equal? (car (cdr (car l))) 'vinv)
         (equal? (car (cdr (car l))) 'ainv)
         (equal? (car (cdr (car l))) 'rinv)
         (equal? (car (cdr (car l))) 'n)
         (equal? (car (cdr (car l))) 'v)
         (equal? (car (cdr (car l))) 'a)
         (equal? (car (cdr (car l))) 'r))
      (cons (car l)
         (solo-relevant-sent (cdr l))))
      (else (solo-relevant-sent (cdr l))))))

;; Transform list of sentences in list of sentences with
;; relevant pairs *only*
(define solo-relevant
  (lambda (lst)
    (cond
      ((null? lst) '())
      (else (cons (solo-relevant-sent (car lst))
        (solo-relevant
        (cdr lst)))))))


;;
;; myflatten for the negation unnecessary nesting
(define myflatten
  (lambda (x)
    (cond
      ((null? x) '())
      ((and
         (pair? (car x))
         (= (length (car x)) 2)) (cons (car x)
         (myflatten (cdr x))))
      (else (myflatten (car x))))))

(define entryflatten
  (lambda (x)
    (cond
      ((null? x) '())
      ((quality-check? (car x))
        (cons (car x) (entryflatten (cdr x))))
      (else (cons (myflatten (car x))
        (entryflatten (cdr x)))))))

(define quality-check?
  (lambda (x)
    (cond
      ((null? x) #t)
      ((and
         (pair? (car x))
         (= (length (car x)) 2))
```

```
      (quality-check? (cdr x)))
      (else #f))))


;;
;; good-dataset?
(define good-dataset?
  (lambda (lop)
    (cond
      ((null? lop) #t)
      ((good-aux? (car lop))
        (good-dataset? (cdr lop)))
      (else #f))))

(define good-aux?
  (lambda (l)
    (cond
      ((null? l) #t)
      ((and
        (pair? (car l))
        (= (length (car l)) 2)
        (or
          (symbol? (car (car l)))
          (number? (car (car l))))
        (or
          (symbol? (car (cdr (car l))))
          (number? (car (cdr (car l))))))
      (good-aux? (cdr l)))
      (else (begin (display l) #f)))))

;;
;;
;; fixy-dataset
(define fixy-dataset
  (lambda (lop)
    (cond
      ((null? lop) #t)
      ((fixy-aux? (car lop)) (cons (car lop)
        (fixy-dataset (cdr lop))))
      (else (cons (fix-the-crap (car lop))
        (fixy-dataset (cdr lop)))))))

;; aux de fixy-dataset
(define fixy-aux?
  (lambda (l)
    (cond
      ((null? l) #t)
      ((and
        (pair? (car l))
        (= (length (car l)) 2)
        (or
          (symbol? (car (car l)))
          (number? (car (car l))))
        (or
```

```
          (symbol? (car (cdr (car l))))
          (number? (car (cdr (car l))))))
      (fixy-aux? (cdr l)))
      (else (begin (display l) #f)))))

;; fix-the-crap
(define fix-the-crap
  (lambda (lst)
    (car lst)))


;; Routine for BUT
(define dealwith-but
  (lambda (lop)
    (cond
      ((null? lop) '())
      ((isthere-but?
        (car lop))
      (if (equal? (caaar lop) 'but)
        (cons (cdr (car lop))
          (dealwith-but (cdr lop)))
        (if (buttylasty? (car lop) (length (car lop)))
          (cons (damebeforebutx (car lop)
            (position-butx (car lop) 1) 1)
          (dealwith-but (cdr lop)))
          (cons (filter-but (car lop))
            (dealwith-but (cdr lop))))))
      (else (cons (car lop)
        (dealwith-but (cdr lop)))))))

;; generic isthere-particle?
(define isthere-particle?
  (lambda (l x)
    (cond
      ((null? l) #f)
      ((equal? (car (car l)) x) #t)
      (else (isthere-particle? (cdr l) x)))))

;; isthere-particle instantiated with "but"
(define isthere-but?
  (lambda (l)
    (isthere-particle? l 'but)))

;; filter out "but"
(define filter-but
  (lambda (l)
    (cond
      ((null? l) '())
      ((equal? (car (car l)) 'but) (cdr l))
      (else (filter-but (cdr l))))))

;; returns TRUE if 'but' is last or before-last
;; in a sentence
;; max=length of list; ls=sentence with
```

```
;; but particle
(define buttylasty?
  (lambda (ls max)
    (cond
      ((null? ls) #f)
      ((or
         (= (position-butx ls 1) max)
         (= (position-butx ls 1) (- max 1))) #t)
      (else #f))))

;; devuleve la posiction en la lista de la particula
;; '(but c)
(define position-butx
  (lambda (ls ct)
    (cond
      ((null? ls) 0)
      ((equal? (caar ls) 'but) ct)
      (else (position-butx (cdr ls) (+ ct 1))))))

;; returns sentence BEFORE particle 'but
;; ls =sentence; limite= donde esta el 'but';
;; ct=contador current position
(define damebeforebutx
  (lambda (ls limite ct)
    (cond
      ((null? ls) '())
      ((= limite ct) (cdr ls))
     (else (cons (car ls)
      (damebeforebutx (cdr ls)
      limite (+ ct 1)))))))


;; Routine for DESPITE
;;
;; isthere-particle instantiated with "despite"
(define isthere-despite?
  (lambda (l)
    (isthere-particle? l 'despite)))

(define filter-despite
  (lambda (l flag)
    (cond
      ((null? l) '())
      ((and
         (equal? (car (car l)) 'despite)
         (not (= flag 0))) '())
      (else (cons (car l)
        (filter-despite (cdr l) (+ flag 1)))))))

(define dealwith-despite
  (lambda (lop)
    (cond
      ((null? lop) '())
      ((isthere-despite? (car lop))
```

```
        (cons (filter-despite (car lop) 0)
        (dealwith-despite (cdr lop))))
      (else (cons (car lop)
        (dealwith-despite (cdr lop)))))))

;; Routine for UNLESS followed by negation
;; If sentence contains "unless" and "unless" is
;; followed by a negative clause,
;; disregard the "unless" clause.
;; i.e. Everyone likes this video, unless he
;; is a sociopath.
;;
;;
;; isthere-particle instantiated with "unless"
;; (works at the sentence level)
(define isthere-unless?
  (lambda (l)
    (isthere-particle? l 'unless)))

;; return the sub-sentence right after the 'unless'
;; particle (works at the sentence level)
(define after-unless
  (lambda (l)
    (cond
      ((null? l) '())
      ((equal? (car (car l)) 'unless) (cdr l))
      (else (after-unless (cdr l))))))

;; return the sub-sentence right before the
;; 'unless' particle (works at the sentence level)
(define before-unless
  (lambda (l sem)
    (cond
      ((or
         (null? l)
         (= sem 1)) '())
      ((equal? (car (car l)) 'unless)
        (before-unless l 1))
      (else (cons (car l)
        (before-unless (cdr l) 0))))))

;; returns #t if the sub-sentence after 'unless'
;; is negative (works at the sentence level)
(define filter-unless
  (lambda (l)
    (cond
      ((null? l) #f)
      ((and
         (isthere-unless? l)
         (not (equal? (after-unless l) '()))
         (not (equal? (before-unless l 0) '())))
      (if (seeifinneg? (after-unless l))
        (before-unless l 0)
        (append (before-unless l 0)
```

```
          (after-unless l))))
      (else l))))

;; returns #t if the given word is in the negative lexicon
(define seeifinneg?
  (lambda (l)
    (cond
      ((null? l) #f)
      ((seeifinneg-aux? (car (car l)) neglex) #t)
      (else (seeifinneg? (cdr l))))))

;; auxiliar function for seeifinneg?
(define seeifinneg-aux?
  (lambda (wrd tabla)
    (cond
      ((null? tabla) #f)
      ((equal? wrd (getword (caar tabla))) #t)
      (else (seeifinneg-aux? wrd (cdr tabla))))))

;; deal with 'unless' (works at the full list of
;; test words level)
(define dealwith-unless
  (lambda (l)
    (cond
      ((null? l) '())
      ((isthere-unless? (car l))
        (cons (filter-unless (car l))
        (dealwith-unless (cdr l))))
      (else (cons (car l)
        (dealwith-unless (cdr l)))))))

;;
;; ROUTINE for MODALS
;;
;; Routine not implemented so far (don't see need for it)
;;
;; ---------------------------------------------------------------
;; rutina that removes (punto punto) and anything else that is not
;; (a,ainv,r,rinv,v,vinv,n,ninv)

(define quita-basura
  (lambda (l)
    (cond
      ((null? l) '())
      (else (cons (quita-basura-aux (car l))
        (quita-basura (cdr l)))))))

;; auxiliar of quita basura
(define quita-basura-aux
  (lambda (lst)
    (cond
      ((null? lst) '())
      ((esbuscado? (car lst)) (cons (car lst)
        (quita-basura-aux (cdr lst))))
```

```
      (else (quita-basura-aux (cdr lst))))))

;; predicado de quita-basura-aux
(define esbuscado?
  (lambda (lmin)
    (cond
      ((null? lmin) #f)
      ((or
         (equal? (car (cdr lmin)) 'v)
         (equal? (car (cdr lmin)) 'vinv)
         (equal? (car (cdr lmin)) 'r)
         (equal? (car (cdr lmin)) 'rinv)
         (equal? (car (cdr lmin)) 'a)
         (equal? (car (cdr lmin)) 'ainv)
         (equal? (car (cdr lmin)) 'n)
         (equal? (car (cdr lmin)) 'ninv)) #t)
      (else #f))))

;; replace '() for 'notermsnvar
(define manejador-x
  (lambda (l)
    (cond
      ((null? l) '())
      ((equal? (car l) '())
        (cons 'notermsnvar (manejador-x (cdr l))))
      (else (cons (car l)
        (manejador-x (cdr l)))))))


;;-------------------------------------------------------------------------
;;
;;-------------------------------------------------------------------------
;;-------------------------------------------------------------------------
;;
;; look-up in lexicons
;;
;; look for word in specified lexicon (WORD Level)
(define look4word
  (lambda (word lex)
    (cond
      ((null? lex) 'wordnotfound)
      ((equal? (getword (caar lex)) word) (car lex))
      (else (look4word word (cdr lex))))))


;; get list of vectors from the lexicon that are in the given
;; sentence (SENTENCE Level):
;; revierte polaridad de los Xinv
(define getwords4sent
  (lambda (sent lex)
    (cond
      ((null? sent) '())
      ((equal? (look4word (car (car sent)) lex)
        'wordnotfound) (cons (car (car sent))
```

```
        (getwords4sent (cdr sent) lex)))
      (else (if (or
        (equal? (car (cdr (car sent))) 'ainv)
        (equal? (car (cdr (car sent))) 'rinv)
        (equal? (car (cdr (car sent))) 'ninv)
        (equal? (car (cdr (car sent))) 'vinv))
      (cons (transformervec
        (look4word (car (car sent)) lex))
        (getwords4sent (cdr sent) lex))
      (cons (look4word (car (car sent)) lex)
        (getwords4sent (cdr sent) lex)))))))

;; generate list of the form ( ((vec1 ... vecn) neg)
;; ((vec1 ... veck) pos) ) :
;; runs at the WHOLE-LIST-OF-SENTENCES Level.
(define getyourbearings
  (lambda (lst)
    (cond
      ((null? lst) '())
      ((equal? (car lst) 'notermsnvar) (cons 'notermsnvar
                          (getyourbearings (cdr lst))))
      ((equal? (getwords4sent (car lst) thelex) '())
              (cons (list 'sentwithnowordsinlex '())
                (getyourbearings (cdr lst))))
      (else (cons (getwords4sent (car lst) thelex)
        (getyourbearings (cdr lst)))))))

;; Change the polarity of a word in Lexicon
(define transformervec
  (lambda (vec)
    (list (list->vector (list (vector-ref (car vec) 0)
                        (vector-ref (car vec) 1)
                        (vector-ref (car vec) 3)
                        (vector-ref (car vec) 2)
                        (vector-ref (car vec) 4)
                        (vector-ref (car vec) 5)
                        (vector-ref (car vec) 6)
                        (vector-ref (car vec) 7)
                        (vector-ref (car vec) 8)))
      (if (equal? (car (cdr vec)) 'pos) 'neg 'pos))))


;;-----------------------------------------------------------------------
;; Obtain list of words NOT in Lexicon
;;-----------------------------------------------------------------------
;; Obtain list of words NOT in Lexicon
;;
;; clean list of NOTFOUNDS: To be called right after notfound-list
(define newwordsforlex
  (lambda (l)
    (cond
      ((null? l) '())
      ((not (or
              (equal? (car l) '())
```

```
            (equal? (car l) 'sentwithnowordsinlex)))
      (cons (car l)
      (newwordsforlex (cdr l))))
      (else (newwordsforlex (cdr l))))))

;; produce list of NOTFOUND words
(define notfound-list
  (lambda (l)
    (cond
      ((null? l) '())
      ((equal? (car l) 'notermsnvar)
        (notfound-list (cdr l)))
      ((contain-symbols? (car l))
        (append (notfound-list-aux (car l))
        (notfound-list (cdr l))))
      (else (notfound-list (cdr l))))))

;; returns #t if there are any symbols in the list
(define contain-symbols?
  (lambda (l)
    (cond
      ((null? l) #f)
      ((pair? (car l)) (contain-symbols? (cdr l)))
      (else #t))))

;; aux function of notfound-list
(define notfound-list-aux
  (lambda (l)
    (cond
      ((null? l) '())
      ((pair? (car l)) (notfound-list-aux (cdr l)))
      (else (cons (car l)
        (notfound-list-aux (cdr l)))))))

;; remove duplicates from list
(define remove-duplicated
  (lambda (lst)
    (if (null? lst)
        '()
        (if (member (car lst) (cdr lst))
            (remove-duplicated (cdr lst))
            (cons (car lst)
              (remove-duplicated (cdr lst)))))))

;; cuantas veces aparece una palabra en la lista
;; llamr con aux function with counter = 0
(define quantasrep
  (lambda (ls ll)
    (cond
      ((null? ls) '())
      (else (cons (list (car ls) (quantasrepaux
        (car ls) ll))
        (quantasrep (cdr ls) ll))))))
```

```
;; calcula las ocurrencias de una palabra y devuelve
;; list (val ocurrencias).
(define quantasrepaux
  (lambda (val ll)
    (cond
      ((null? ll) 0)
      ((equal? val (car ll))
        (+ 1 (quantasrepaux val (cdr ll))))
      (else (quantasrepaux val (cdr ll))))))

;; pick from the list those pairs (word occr)
;; where occr >= val
(define getthose>val
  (lambda (val l)
    (cond
      ((null? l) '())
      ((>= (car (cdr (car l))) val) (cons (car l)
        (getthose>val val (cdr l))))
      (else (getthose>val val (cdr l))))))

;; remove repeated pairs (word num_occurs)
(define paresdupno
  (lambda (l)
    (cond
      ((null? l) '())
      ((member (car l) (cdr l)) (paresdupno (cdr l)))
      (else (cons (car l) (paresdupno (cdr l)))))))



;-------------------------------------------------------------------------
;;END of  Obtain list of words NOT in Lexicon
;;-------------------------------------------------------------------------
;;
;;-------------------------------------------------------------------------
;;-------------------------------------------------------------------------

;; return number of occurrences in list of sentences that is not
;;  the expected PoS
(define cuantos-cada-weird
  (lambda (lst)
    (cond
      ((null? lst) 0)
      ((or
         (equal? (getpos (car lst)) 'n)
         (equal? (getpos (car lst)) 'v)
         (equal? (getpos (car lst)) 'a)
         (equal? (getpos (car lst)) 'nopostag)
         (equal? (getpos (car lst)) 'r))
      (cuantos-cada-weird (cdr lst)))
      (else (cons (getpos (car lst))
        (cuantos-cada-weird (cdr lst)))))))

;; return number of occurrences in list of sentences of
```

```
;; a particular PoS
(define cuantos-cada-pos
  (lambda (lst thepos)
    (cond
      ((null? lst) 0)
      ((equal? (getpos (car lst)) thepos)
        (+ 1 (cuantos-cada-pos
        (cdr lst) thepos)))
      (else (cuantos-cada-pos (cdr lst) thepos)))))

;; return number of occurrences in list of sentences
;; of a particular PoS
(define pos-s
  (lambda (lst)
    (cond
      ((null? lst) '())
      ((equal? (getpos (car lst)) 's)
        (cons (car lst) (pos-s (cdr lst))))
      (else (pos-s (cdr lst))))))


;; call with list and min=99
(define mininlist
  (lambda (l min)
    (cond
      ((null? l) min)
      ((< (car l) min)
        (mininlist (cdr l) (car l)))
      (else (mininlist (cdr l) min)))))

;; call with list and max=-99
(define maxinlist
  (lambda (l max)
    (cond
      ((null? l) max)
      ((> (car l) max) (maxinlist (cdr l) (car l)))
      (else (maxinlist (cdr l) max)))))

;;-------------------------------------------------------------------------------
;;
;; l=lista de valores devueltos por rutinas de rules, counter=mark
;; position in list,
;; call first  with 1,
;; val=value we want to compare for in terms of length of list of values
(define vacionopuede
  (lambda (l counter val)
    (cond
      ((null? l) '())
      ((and
         (pair? (car l))
         (>= (length (car l)) val)) (vacionopuede (cdr l)
         (+ counter 1) val))
      (else (cons (list counter (car l)) (vacionopuede (cdr l)
        (+ counter 1) val))))))
```

```
;; ------------------------------------------------------------------------
;; BUSCA en SentiWordNet palabras not found in original Lexicon
;;------------------------------------------------------------------------
;;
(define newwordys
  (lambda (lwords lex)
    (cond
      ((null? lwords) '())
      ((newwordys-aux? (car lwords) lex)
        (cons (newwordys-aux-cons (car lwords) lex)
                    (newwordys (cdr lwords) lex)))
      (else (newwordys (cdr lwords) lex)))))

(define newwordys-aux?
  (lambda (word lex)
    (cond
      ((null? lex) #f)
      ((equal? (vector-ref (car lex) 0) word) #t)
      (else (newwordys-aux? word (cdr lex))))))

(define newwordys-aux-cons
  (lambda (word lex)
    (cond
      ((null? lex) 'notfound)
      ((equal? (vector-ref (car lex) 0) word) (car lex))
      (else (newwordys-aux-cons word (cdr lex))))))

;;
(define limpia-molleja
  (lambda (l)
    (cond
      ((null? l) '())
      (else (cons (vector-ref (car l) 0)
        (limpia-molleja (cdr l)))))))

(define find-stuff
  (lambda (l1 l2)
    (cond
      ((null? l1) '())
      ((member (car l1) l2) (find-stuff (cdr l1) l2))
      (else (cons (car l1) (find-stuff (cdr l1) l2))))))

;; list pos-scores of vecs
(define psrels
  (lambda (l)
    (cond
      ((null? l) '())
      ((and
         (number? (getposscore (caar l)))
         (not (= (getposscore (caar l)) 0)))
      (cons (getposscore (caar l))
         (psrels (cdr l))))
      (else (psrels (cdr l))))))
```

```
;;
;; Convert len=6 vectors augmented with SentiWordNet to
;; len=9 vectors in list
;; of form ( (v1 sor)...(vn sor) )
(define morfear
  (lambda (lvecs)
    (cond
      ((null? lvecs) '())
      (else (cons (make-morfear (car lvecs))
        (morfear (cdr lvecs)))))))

(define make-morfear
  (lambda (vec)
    (list (list->vector (list (getword vec)
                        (getpos vec)
                        (getposscore vec)
                        (getnegscore vec)
                        (getobjscore vec)
                        'nocsor
                        'nomaxdist
                        'nomindist
                        1)) 'obj)))


;;-------------------------------------------------------------------------
;; CREAR new lexs with format (vector pos/neg/obj)
;;-------------------------------------------------------------------------
;; lst = lexicon; label=pos/neg/obj
(define creanewlexman
  (lambda (lst label)
    (if (null? lst)
        '()
        (cons (list (car lst) label) (creanewlexman (cdr lst) label)))))

;;-------------------------------------------------------------------------
;; LEXICON FORMAT SAMPLE at Initialisation time
;;-------------------------------------------------------------------------
;; initialise the lexicons. They should be loaded from the variable
;;  'readbuffer' generated by read-file
;;(define theneglex '())
;;(define theposlex '())
;;(define dict '( #(two-faced nopostag nopsc nonsc nocobj
;;  nocsor nomaxdist nomindist 0)
;;                #(two-faces nopostag nopsc nonsc nocobj
;;  nocsor nomaxdist nomindist 0)
;;                #(abnormal a 0.0 0.75 0.25 nocsor
;;   nomaxdist nomindist 0)) )
;;

;;-----------------------------------------------------------------------
;; Orestes Appel - LexEditor
;;-----------------------------------------------------------------------
;;=== Initial CALL TO THE SYSTEM by the User ===
;;=============================================
```

```
(define lexed
  (lambda ()
    (begin
      (welcome)
      (procesador oldpanalex 1))))
;;
;; initial header when entering the system
;;
(define welcome
  (lambda ()
    (display "------------------------------------------------------")
    (newline)
    (display "---␣␣␣␣Welcome␣to␣the␣Lexicon␣Editor␣System␣␣␣␣␣␣␣---")
    (newline)
    (display "------------------------------------------------------")
    (newline)
    (display "A␣pair␣(Vector␣Orientation)␣will␣be␣displayed.
␣␣␣␣␣␣Please␣enter␣the␣desired␣Orientation␣value␣and␣hit␣ENTER.")
    (newline)
    (display "Enter␣the␣word␣END␣when␣done.")
    (newline)))

;;
;; initial dialog to decide the proper way of entering data
;;
(define procesador
  (lambda (l count)
    (if (null? l)
        (begin
          (newline)
          (display "␣-----------------------------------------------")
          (newline)
          (display "␣---␣Thanks␣for␣using␣the␣system.
␣␣␣␣␣␣␣␣␣␣␣␣Have␣a␣nice␣day.␣---")
          (newline)
          (display "␣-----------------------------------------------")
          (newline))
        (begin
          (newline)
          (display (list count (car l)))
          (newline)
          (display "Your␣selection==>␣")
          (let ((sel (read)))
            (if (or (equal? sel 'end)
                    (equal? sel 'END))
                (begin
                  (newline)
                  (display "␣-----------------------------------------------")
                  (newline)
                  (display "␣---␣Thanks␣for␣using␣the␣system.
␣␣␣␣␣␣␣␣␣␣␣␣␣␣Have␣a␣nice␣day.␣---")
                  (newline)
                  (display "␣-----------------------------------------------")
                  (newline))
```

```
                (begin
                  (set! newpanalex (cons (patch-vec
                    (caar l) sel) newpanalex))
                  (newline)
                  (procesador (cdr l) (+ count 1)))))))))

(define patch-vec
  (lambda (v so)
    (list v (if (equal? so 'p)
                'pos
                (if (equal? so 'n)
                    'neg
                    (if (equal? so 'o)
                        'obj
                        'del))))))

;;
(define goodpanalex?
  (lambda (l)
    (cond
      ((null? l) #t)
      ((and
         (= (length (car l)) 2)
         (vector? (caar l))
         (= (vector-length (caar l)) 9)
         (symbol? (car (cdr (car l))))
         (or
           (equal? (car (cdr (car l))) 'pos)
           (equal? (car (cdr (car l))) 'neg)
           (equal? (car (cdr (car l))) 'del)
           (equal? (car (cdr (car l))) 'obj)))
       (goodpanalex? (cdr l)))
       (else #f))))

(define tellmebro
  (lambda (l)
    (cond
      ((null? l) '())
      ((or
         (equal? (car (cdr (car l))) 'pos)
         (equal? (car (cdr (car l))) 'neg)
         (equal? (car (cdr (car l))) 'obj))
       (tellmebro (cdr l)))
       (else (cons (car l)
         (tellmebro (cdr l)))))))

(define repetido?
  (lambda (l)
    (cond
      ((null? l) '())
      ((vecmember (car l) (cdr l)) (cons (car l)
        (repetido? (cdr l))))
      (else (repetido? (cdr l))))))
```

```
(define vecmember
  (lambda (ele ls)
    (cond
      ((null? ls) #f)
      ((equal? (getword (car ele))
        (getword (caar ls))) #t)
      (else (vecmember ele (cdr ls))))))

(define rightvecky
  (lambda (lst)
    (if (null? lst)
        '()
        (cons (buildmevecky (car lst))
          (rightvecky (cdr lst))))))

(define buildmevecky
  (lambda (lst)
    (list (list->vector (list (getword (car lst))
                              (getpos (car lst))
                              (getposscore (car lst))
                              (getnegscore (car lst))
                              (getobjscore (car lst))
                              'nocsor
                              'nomaxdist
                              'nomindist
                              1))
          (car (cdr lst)))))

(define bveckyfwords
  (lambda (l)
    (if (null? l)
        '()
        (cons (bveckyfwords-aux (car l))
          (bveckyfwords (cdr l))))))

(define bveckyfwords-aux
  (lambda (wrd)
    (list (list->vector (list wrd
                              'nopostag
                              'nopsc
                              'nonsc
                              'nocobj
                              'nocsor
                              'nomaxdist
                              'nomindist
                              1))
          'obj)))

(define sacametenegs
  (lambda (lst)
    (if (null? lst)
        '()
        (if (null? (sacametenegs-x (car lst)))
            (sacametenegs (cdr lst))
```

```
            (cons (sacametenegs-x (car lst))
              (sacametenegs (cdr lst)))))))


(define sacameteposs
  (lambda (lst)
    (if (null? lst)
        '()
        (if (null? (sacameteposs-x (car lst)))
            (sacameteposs (cdr lst))
            (cons (sacameteposs-x (car lst))
              (sacameteposs (cdr lst)))))))

 (define sacameteobjs
  (lambda (lst)
    (if (null? lst)
        '()
        (if (null? (sacameteobjs-x (car lst)))
            (sacameteobjs (cdr lst))
            (cons (sacameteobjs-x (car lst))
              (sacameteobjs (cdr lst)))))))

(define sacameteposs-x
  (lambda (l)
    (if (equal? (car (cdr l)) 'pos)
        l
        '())))

(define sacametenegs-x
  (lambda (l)
    (if (equal? (car (cdr l)) 'neg)
        l
        '())))

(define sacameteobjs-x
  (lambda (l)
    (if (equal? (car (cdr l)) 'obj)
        l
        '())))

;; agrega lable pos/neg/obj a lista de vectores
;; retornados de SentiWordNet
;; Pasa l=lista de new vecs; label=pos/neg/obj
(define agregalabel
  (lambda (l label)
    (if (null? l)
        '()
        (cons (list (car l) label)
          (agregalabel (cdr l) label)))))

;; compara lista de palabras = (p1...pn) contra
;; new lexicon ((v label)...)
;; y devuleve palabras que no estan en el
;; nuevo lexicon
```

```
(define damelos
  (lambda (l lex)
    (if (null? l)
        '()
        (if (not (damelos-aux? (car l) lex))
            (cons (damelos2 (car l) lex)
              (damelos (cdr l) lex))
            (damelos (cdr l) lex)))))

(define damelos-aux?
  (lambda (wrd lex)
    (cond
      ((null? lex) #f)
      ((equal? (getword (caar lex)) wrd) #t)
      (else (damelos-aux? wrd (cdr lex))))))

(define damelos2
  (lambda (wrd lex)
    (cond
      ((null? lex) wrd)
      ((equal? (getword (caar lex)) wrd) '())
      (else (damelos2 wrd (cdr lex))))))

;;-------------------------------------------------------
;; take 2 lists of palabras and returns a list with
;; p1-words not in p2
(define mamachicha
  (lambda (p1 p2)
    (cond
      ((null? p1) '())
      ((member (car p1) p2) (mamachicha
        (cdr p1) p2))
      (else (cons (car p1) (mamachicha
        (cdr p1) p2))))))

;; ve si las palabras en una lista estan repetidas y
;; retorna such a list
(define whatthereps
  (lambda (l)
    (cond
      ((null? l) '())
      ((member (car l) (cdr l)) (cons (car l)
        (whatthereps (cdr l))))
      (else (whatthereps (cdr l))))))

;; build a lex-vector from a list of words (w1,...,wn)
(define gimmevecky
  (lambda (l)
    (if (null? l)
        '()
        (cons (gimmevecky-aux (car l))
          (gimmevecky (cdr l))))))

;; aux for gimmevecky
```

```
(define gimmevecky-aux
  (lambda (wrd)
    (list->vector (list wrd
                        'nopostag
                        'nopsc
                        'nonsc
                        'nocobj
                        'nocsor
                        'nomaxdist
                        'nomindist
                        1))))

;;-----------------------------------------------------------------
;;
;;-----------------------------------------------------------------
;; Compare list of words with vec from SentiWordNet and return
;; a list of SentiWordNet vecs for those words found in vec
;;
(define morevecstouse
  (lambda (l lvecs)
    (if (null? l)
        '()
        (cons (morevecstouse-aux (car l) lvecs)
          (morevecstouse (cdr l) lvecs)))))

;; aux function for morevecstouse
(define morevecstouse-aux
  (lambda (wrd vecs)
    (cond
      ((null? vecs) 'notfound)
      ((equal? (getword (car vecs)) wrd) (car vecs))
      (else (morevecstouse-aux wrd (cdr vecs))))))

;; remove particle 'notfound from a list of vectors
;; and notfoundS
(define remnots
  (lambda (l)
    (if (null? l)
        '()
        (if (vector? (car l))
            (cons (car l) (remnots (cdr l)))
            (remnots (cdr l))))))


;;-----------------------------------------------------------------
;; transform list of vectors in list of compliant vectors
(define trantranvecs
  (lambda (l)
    (if (null? l)
        '()
        (cons (trantranvecs2 (car l))
          (trantranvecs (cdr l))))))

;; aux of trantranvecs
```

```
(define trantranvecs2
  (lambda (vec)
    (list->vector
      (list (getword vec)
            (getpos vec)
            (getposscore vec)
            (getnegscore vec)
            (getobjscore vec)
            'nocsor
            'nomaxdist
            'nomindist
            1))))

;; Transform list of vectors in list of vectors and label
;; (v1...vn) ---> ( (v1 label) ... (vN label) )
(define tranveckyx
  (lambda (lvecs)
    (if (null? lvecs)
        '()
        (cons (list (car lvecs) 'obj)
          (tranveckyx (cdr lvecs))))))

;; Remove from list vectors lablled as 'del'
(define remdelcases
  (lambda (lvecs)
    (cond
      ((null? lvecs) '())
      ((equal? (car (cdr (car lvecs))) 'del)
        (remdelcases (cdr lvecs)))
      (else (cons (car lvecs)
        (remdelcases (cdr lvecs)))))))

;; todos las listas (v1 label) sond de label=xxx
(define todosbuenoschicos
  (lambda (ls xxx)
    (cond
      ((null? ls) #t)
      ((equal? (car (cdr (car ls))) xxx)
        (todosbuenoschicos (cdr ls) xxx))
      (else #f))))

;; Remove from list vectors lablled as 'pos'
(define remposcases
  (lambda (lvecs)
    (cond
      ((null? lvecs) '())
      ((equal? (car (cdr (car lvecs))) 'pos)
        (remposcases (cdr lvecs)))
      (else (cons (car lvecs)
        (remposcases (cdr lvecs)))))))

;; Remove from list vectors lablled as 'neg'
(define remnegcases
  (lambda (lvecs)
```

```
    (cond
      ((null? lvecs) '())
      ((equal? (car (cdr (car lvecs))) 'neg)
        (remnegcases (cdr lvecs)))
      (else (cons (car lvecs)
        (remnegcases (cdr lvecs)))))))

;; Remove from list vectors lablled as 'obj'
(define remobjcases
  (lambda (lvecs)
    (cond
      ((null? lvecs) '())
      ((equal? (car (cdr (car lvecs))) 'obj)
        (remobjcases (cdr lvecs)))
      (else (cons (car lvecs)
        (remobjcases (cdr lvecs)))))))

;;=============================================================
;;=== validar resultados de "getyourbearings"               ===
;;=============================================================
;; return list of number of sentences for which
;; there is not even one word in the lexicon
(define enrollo
  (lambda (ls ct)
    (cond
      ((null? ls) '())
      ((atom? (car ls)) (cons ct
        (enrollo (cdr ls) (+ ct 1))))
      (else (enrollo (cdr ls) (+ ct 1))))))

;; retorna #t si una lista representando una
;; oracion es un singleton
(define essingleton?
  (lambda (l)
    (if (= (length l) 1) #t #f)))

;; retorna lista de numeros de oraciones que son singletons
(define lossingletons
  (lambda (ls ct)
    (if (null? ls)
        '()
        (if (essingleton? (car ls))
            (cons ct (lossingletons (cdr ls) (+ ct 1)))
            (lossingletons (cdr ls) (+ ct 1))))))

;; por lo menos un vector
(define por-unvector
  (lambda (ls)
    (cond
    ((null? ls) '())
    ((porlomenosuno? (car ls)) (por-unvector (cdr ls)))
    (else (cons (car ls) (por-unvector (cdr ls)))))))

;; auxiliar de por-unvector
```

```
(define porlomenosuno?
  (lambda (l)
    (cond
    ((null? l) #f)
    ((and
       (pair? (car l))
       (vector? (caar l))
       (symbol? (car (cdr (car l))))) #t)
    (else (porlomenosuno? (cdr l))))))

;;============================================================
;;=== Take list of atomic words and make it
;;    into list of (( v1 label) ... ) ===
;;============================================================
(define construlistvec
  (lambda (ls label)
    (if (null? ls)
        '()
        (cons (constru-aux (car ls) label)
          (construlistvec (cdr ls) label)))))

;; auxiliar function of construlistvec
(define constru-aux
  (lambda (wrd label)
    (list (list->vector
      (list wrd 'nopostag 'nonsc 'nopsc 'nocobj
        'nocsor 'nomaxdist 'nomindist 3)) label)))

;;==================================
;;==== Manage duplicates ==========
;;==================================
;; remove duplicates from list
(define remove-duplicated-lat
  (lambda (lst)
    (if (null? lst)
        '()
        (if (member (car lst) (cdr lst))
            (remove-duplicated-lat (cdr lst))
            (cons (car lst)
              (remove-duplicated-lat (cdr lst)))))))

;; compare two list and return words in both lists
;; input is l1, l2 = ( (vector label) ... (vector label) )
(define myintersect
  (lambda (l1 l2)
    (cond
      ((null? l1) '())
      (else (cons (myintersect-aux (car l1) l2)
        (myintersect (cdr l1) l2))))))

;; auxiliar function of myintersect
(define myintersect-aux
  (lambda (l l2)
    (if
```

```
      (null? l2) '()
      (if (memberspecial (getword (car l)) l2)
          (cons l (myintersect-aux l (cdr l2)))
          'none))))


;; memberspecial: sasa
(define memberspecial
  (lambda (wrd ls)
    (cond
      ((null? ls) #f)
      ((equal? wrd (getword (caar ls))) #t)
      (else (memberspecial wrd (cdr ls))))))

;; crear lista de palabras a partir de
;; diccionario-like files like atoms
;; despojar ( (v1 label) ... (v2 label) )
;;  ---> (wrd1 wrd2)
;;
(define despojar
  (lambda (ls)
    (if (null? ls)
        '()
        (cons (vector-ref (caar ls) 0)
          (despojar (cdr ls))))))

;; Function reciproca de  remove duplicates from list
(define remove-duplicated-lat-rec
  (lambda (lst)
    (if (null? lst)
        '()
        (if (not (member (car lst) (cdr lst)))
            (remove-duplicated-lat-rec (cdr lst))
            (cons (car lst)
              (remove-duplicated-lat-rec
                (cdr lst)))))))

;;-----------------------------------------------------------------
;; Orestes Appel
;;-----------------------------------------------------------------
;;-----------------------------------------------------------------
;; Start module for I/O and syntactical checking [miio.ss]
;;-----------------------------------------------------------------
;;================================  Start of program ===============
;;
;;---------------------------Global Bindings ----------------------
(define readbuffer '())
;;
;;------------------------- Begin Input track ---------------------
;;
;; READ all info held in the file containing the query
;;
(define read-file
  (lambda (name)
    (let ([p (open-input-file name)])
```

```
      (set! readbuffer (read p))
      (close-input-port p))))

;;------------------------------ End input track ---------------------
;;-----------------------------Begin Output track ---------------------
;;
;;
;; Write down to a file called "ChooseName"
;; the information processed
;;
(define downto-file
  (lambda (val name)
    (begin
      (let ([p (open-output-file name)])
              (write val p)
              (close-output-port p))
      (newline)
      (display "Writing completed.")
      (newline))))

;;
;; verifies if a file exists or not
;;
(define existfile?
  (lambda (f)
    (file-exists? f)))

;;------------------------------ End output track ------------------
;;------------------------------ Auxiliary track -------------------

;;
;; reverse a given list 'l'
;;
(define reverse
  (lambda (l)
    (cond
     [(null? l) '()]
     [else (snoc (car l) (reverse (cdr l)))])))

;;
;; inverse of CONS
;;
(define snoc
  (lambda (x ls)
    (append ls (list x))))



;;------------------------------ End auxiliary track ------
;; Para listas = ( (a num) (b num) (c num))
(define eleval>1
  (lambda (lista)
    (cond
      ((null? lista) '())
```

```
      ((> (car (cdr (car lista))) 1.00)
        (begin
          (display (car lista))
          (newline)
          (cons (car lista) (eleval>1 (cdr lista)))))
      (else (begin
        (display (car lista))
        (eleval>1 (cdr lista)))))))

(define mirar
  (lambda (lista cont)
    (cond
      ((null? lista) (display cont))
      (else
        (begin
        (display (car lista))
        (newline)
        (mirar (cdr lista)
          (+ cont 1)))))))

(define get-distances
  (lambda (lwords ldists)
    (cond
      ((null? lwords) '())
      (else (cons (sweepwords (car lwords) ldists)
        (get-distances (cdr lwords) ldists))))))

(define sweepwords
  (lambda (word ldists1)
    (cond
      ((null? ldists1) 'notfound)
      ((equal? word (car (car ldists1)))
        (list word (car (cdr (car ldists1)))))
      (else (sweepwords word (cdr ldists1))))))

(define allpairs?
  (lambda (list)
    (cond
      ((null? list) #t)
      (else (if (pair? (car list))
                (allpairs? (cdr list))
                #f)))))

(define atomlist?
  (lambda (l)
    (cond
      ((null? l) #t)
      (else (if (atom? (car l))
                (atomlist? (cdr l))
                #f)))))

(define countwords
  (lambda (l1 l2 cont)
    (cond
```

```
      ((null? l1) 0)
      (else (if (isin? (car l1) l2)
                (countwords ((cdr l2) l2
                  (+ cont 1)))
                (countwords (cdr l2)
                  l2 cont))))))

(define isin?
  (lambda (word l2)
    (cond
      ((null? l2) #f)
      (else (if (equal? word (car l2))
                #t
                (isin? word (cdr l2)))))))

(define remove-notfounds
  (lambda (l)
    (cond
      ((null? l) '())
      ((equal? (car l) 'notfound)
        (remove-notfounds (cdr l)))
      (else (cons (car l)
        (remove-notfounds (cdr l)))))))

;; ojo
(define (build-list name)
  (let ((port (open-input-file name)))
    (build-list-helper port 1)
    (close-input-port port)
    'done))

(define (build-list-helper port conta) ;; 1st version
  (let ((stuff (get-line port)))
    (if (eof-object? stuff)
        'done
        (begin (display stuff)
               (newline)
               (set! pupu (snoc
                (list conta stuff) pupu))
               (build-list-helper port
          (+ conta 1))))))
```

C.6 Dictionary-building Code

```
;;=========================================
;;========= generate dicts NEG/POS words ====
;;=========================================
(load "/Users/orestesappel/Desktop/SchemeLibrary/Code/
␣␣␣␣␣␣␣␣␣␣␣␣sorcalc.ss")
(load "/Users/orestesappel/Desktop/SchemeLibrary/Code/
␣␣␣␣␣␣␣␣␣␣␣␣sorcalcv2.ss")
(load "/Users/orestesappel/Desktop/SchemeLibrary/Code/
␣␣␣␣␣␣␣␣␣␣␣␣sorcalcv3.ss")
;;
```

```
(define gendict-main
  (lambda ()
    (corri-dict)))
```

Appendix D

Data Preparation & Processing

The following text is based on the assumption that the programming language Python (`https://www.python.org/`) is available, as well as the NLTK [25] toolkit for Natural Languages (`http://www.nltk.org/`).

Prof. Christopher Potts, Associate Professor of Linguistics, Director of CSLI at Stanford University, Department of Linguistics `http://web.stanford.edu/~cgpotts/`, has produced code and documentation for a number of manipulations using WordNet (`http://wordnet.princeton.edu/` [83, 147] and SentiWordNet (`http://sentiwordnet.isti.cnr.it/`) [79, 80]. The following is Prof. Potts link for complete details of the code and additional explanations `http://compprag.christopherpotts.net/wordnet.html` and for information about the teaching activities of Prof. Potts `http://web.stanford.edu/~cgpotts/teaching.html`.

Prof. Potts have written a Python interface to SentiWordNet. By placing this in the same directory as one's SentiWordNet source file (restricted link) and then entering the following commands, like follows:

$python > from sentiwordnet import SentiWordNetCorpusReader, SentiSynset$
$python > swn\_filename = `SentiWordNet\_3.0.0\_20100705.txt'$
$python > swn = SentiWordNetCorpusReader(swn\_filename)$

it is possible to access SentiWordNet to extract critical information resident there. Below a sample of the SentiWorldNet interface written by Prof. Potts.

Our thought process in order to build our Opinion Lexicon, was to used as a base the Lexicons (Positive and Negative) produced by Prof. Bing Liu (REF) and enrich it with *synsets* data held in SentiWordNet with scores for positive and/or negative meanings for a number of existing words. Here is the structure of a Synset (SentiSynset) in SentiWordNet:

POS	**ID**	**PosScore**	**NegScore**	**SynsetTerms**	**Gloss...**
a	00001740	0.125	0	able#1	(usually followed by 'to') having the ...

POS can take five possible values:

a = Adjective
n = Noun
v = Verb
r = Adverb

s = Adjective Satellite

The output of the SentiWordNet Viewer written by Prof. Potts has the following format:

Word.PoS.NoOfSynonyms.PositiveScore.NegativeScore
snowmobile.n.01 0.0 0.0
fortunate.s.02 0.875 0.0
temperature.n.02 0.0 0.25
summer.n.02 0.0 0.0
whirring.s.01 0.0 0.0 ...

This format needs to be transformed into something friendlier to manipulate. As such, this is the process we followed:

1. Print to a file all polarity-augmented *Synsets* (sets of cognitive synonyms, each expressing a distinct concept) in WordNet 3.0 (`http://wordnet.princeton.edu`) that are currently available in SentiWordNet 3.0 (`http://sentiwordnet.isti.cnr.it`) and build a *list*
2. Transform the above (Step 1 outcome)into a type *list of vectors* in the Scheme Programming Language (`http://en.wikipedia.org/wiki/Scheme_(programming_language)`) [73].
3. Generate a list of words with *positive* meanings starting with Prof. Bing Liu et al. work. According to Bing Liu "this list was compiled over many years starting from our first paper (Minqing Hu and Bing Liu. 'Mining and summarizing customer reviews.'" [102].
4. Generate a list of words with *negative* meanings starting with Prof. Bin Liu et al. work [102].
5. Using the list from Step 2 as a source, manipulate the lists from Step 3 and Step 4, replacing in them the words existing in the list from Step 2 with their respective SentiSynsets
6. Obtain an *Opinion Lexicon* that for pragmatic and computational purposes will encompass two lists of vectors, each representing a SentiSynset. One list for positive meanings and another one for negative meanings.

D.1 SentiWordNet Interface

```
#!/usr/bin/env python

"""
Interface to SentiWordNet using the NLTK WordNet classes.

---Chris Potts
"""

import re
import os
import sys
import codecs
```

```
try:
    from nltk.corpus import wordnet as wn
except ImportError:
    sys.stderr.write("Couldn't find an NLTK installation.\n")
    sys.exit(2)

#####################################################################

class SentiWordNetCorpusReader:
    def __init__(self, filename):
        """
        Argument:
        filename -- the name of the text file containing the
                    SentiWordNet database
        """
        self.filename = filename
        self.db = {}
        self.parse_src_file()

    def parse_src_file(self):
        lines = codecs.open(self.filename, "r", "utf8").read().splitlines()
        lines = filter((lambda x : not re.search(r"^\s*#", x)), lines)
        for i, line in enumerate(lines):
            fields = re.split(r"\t+", line)
            fields = map(unicode.strip, fields)
            try:
                pos, offset, pos_score, neg_score, synset_terms, gloss = fields
            except:
                sys.stderr.write("Line %s formatted incorrectly: %s\n" % (i, line))
            if pos and offset:
                offset = int(offset)
                self.db[(pos, offset)] = (float(pos_score), float(neg_score))

    def senti_synset(self, *vals):
        if tuple(vals) in self.db:
            pos_score, neg_score = self.db[tuple(vals)]
            pos, offset = vals
            synset = wn._synset_from_pos_and_offset(pos, offset)
            return SentiSynset(pos_score, neg_score, synset)
        else:
            synset = wn.synset(vals[0])
            pos = synset.pos
            offset = synset.offset
            if (pos, offset) in self.db:
                pos_score, neg_score = self.db[(pos, offset)]
                return SentiSynset(pos_score, neg_score, synset)
            else:
                return None

    def senti_synsets(self, string, pos=None):
        sentis = []
        synset_list = wn.synsets(string, pos)
        for synset in synset_list:
```

```
            sentis.append(self.senti_synset(synset.name))
        sentis = filter(lambda x : x, sentis)
        return sentis

    def all_senti_synsets(self):
        for key, fields in self.db.iteritems():
            pos, offset = key
            pos_score, neg_score = fields
            synset = wn._synset_from_pos_and_offset(pos, offset)
            yield SentiSynset(pos_score, neg_score, synset)

#####################################################################

class SentiSynset:
    def __init__(self, pos_score, neg_score, synset):
        self.pos_score = pos_score
        self.neg_score = neg_score
        self.obj_score = 1.0 - (self.pos_score + self.neg_score)
        self.synset = synset

    def __str__(self):
        """Prints just the Pos/Neg scores for now."""
        s = ""
        s += self.synset.name + "\t"
        s += "PosScore: %s\t" % self.pos_score
        s += "NegScore: %s" % self.neg_score
        return s

    def __repr__(self):
        return "Senti" + repr(self.synset)

#####################################################################

if __name__ == "__main__":
    """
    If run as

    python sentiwordnet.py

    and the file is in this directory, send all of the SentiSynSet
    name, pos_score, neg_score trios to standard output.
    """
    SWN_FILENAME = "SentiWordNet_3.0.0_20100705.txt"
    if os.path.exists(SWN_FILENAME):
        swn = SentiWordNet(SWN_FILENAME)
        for senti_synset in swn.all_senti_synsets():
            print senti_synset.synset.name, senti_synset.pos_score,
                senti_synset.neg_score
```

D.2 NLP manipulations

The following Annex includes bits of code written in Python with the purpose of:

- Tagging sentences according to their grammatical type (nouns, verbs, adjectives, etc.)

- Parsing the sentences in a proper fashion getting them ready for further analysis and processing
- Expanding contractions, like “I’ll” into “I will”, “she’s” into “she is”, etc., as explained in the code

The code provided below, written in Python is available through the book ‘Natural Language Processing with Python’ by Bird [26] and the Perkins’ book [173], ‘Python Text Processing with NLTK 2.0 Cookbook’ that provides interesting pieces of Python code oriented toward performing a variety of NLP tasks.

Note: this code has *not* be written by the author of this PhD Thesis, but produced and distributed by the sources described in the previous paragraph. However, the modifications and adjustments required to generate the products and outputs required for our research are 100% mine. Below, the aforementioned code.

```
import re, csv, yaml
from nltk.corpus import wordnet
from nltk.metrics import edit_distance

#####################################################
## Replacing Words Matching Regular Expressions ##
#####################################################

replacement_patterns = [
        (r'won\'t', 'will not'),
        (r'WON\'T', 'will not'),
        (r'DON\'T', 'do not'),
        (r'don\'t', 'do not'),
        (r'can\'t', 'can not'),
        (r'Can\'t', 'can not'),
        (r'CAN\'T', 'can not'),
        (r'didn\'t', 'did not'),
        (r'DIDN\'T', 'did not'),
        (r'didnt', 'did not'),
        (r'DIDNT', 'did not'),
        (r'i\'d', 'i would'),
        (r'i\'m', 'i am'),
        (r'I\'m', 'i am'),
        (r'ain\'t', 'is not'),
        (r'(\w+)\'ll', '\g<1> will'),
        (r'(\w+)n\'t', '\g<1> not'),
        (r'(\w+)\'ve', '\g<1> have'),
        (r'(\w+t)\'s', '\g<1> is'),
        (r'(\w+)\'re', '\g<1> are'),
        (r'(\w+)\'d', '\g<1> would'),
        (r'\'em', 'them'),
        (r'ol\'', 'old'),
        (r'ole\'', 'old'),
        (r'cant', 'can not'),
        (r'there\'s', 'there is'),
        (r'there\'re', 'there are'),
        (r'who\'s', 'who is'),
        (r'here\'s', 'here is'),
        (r'where\'s', 'where is'),
        (r'he\'s', 'he is'),
        (r'she\'s', 'she is'),
```

```
        (r'isn\'t', 'is not'),
        (r'aren\'t', 'are not'),
        (r'(\w+)\'ed', '\g<1>ed'),
        (r'(\w+)\'d', '\g<1>d'),
        (r'(\w+)\'s', '\g<1>s'),
        (r'(\w+)\'n', '\g<1>n'),
        (r'(\w+)\'', '\g<1>'),
        (r'\'(\w+)', '\g<1>'),
        (r'\'', ' '),
]

class RegexpReplacer(object):
        """ Replaces regular expression in a text.
        >>> replacer = RegexpReplacer()
        >>> replacer.replace("can't is a contraction")
        'cannot is a contraction'
        >>> replacer.replace("I should've done that thing I didn't do")
        'I should have done that thing I did not do'
        """
        def __init__(self, patterns=replacement_patterns):
                self.patterns = [(re.compile(regex), repl) for (regex, repl) in patterns]

        def replace(self, text):
                s = text

                for (pattern, repl) in self.patterns:
                        (s, count) = re.subn(pattern, repl, s)

                return s

#####################################
## Replacing Repeating Characters ##
#####################################

class RepeatReplacer(object):
        """ Removes repeating characters until a valid word is found.
        >>> replacer = RepeatReplacer()
        >>> replacer.replace('looooove')
        'love'
        >>> replacer.replace('oooooh')
        'ooh'
        >>> replacer.replace('goose')
        'goose'
        """
        def __init__(self):
                self.repeat_regexp = re.compile(r'(\w*)(\w)\2(\w*)')
                self.repl = r'\1\2\3'

        def replace(self, word):
                if wordnet.synsets(word):
                        return word

                repl_word = self.repeat_regexp.sub(self.repl, word)
```

```
                if repl_word != word:
                        return self.replace(repl_word)
                else:
                        return repl_word

########################################
## Spelling Correction with Enchant ##
########################################

class SpellingReplacer(object):
        """ Replaces misspelled words with a likely suggestion based on shortest
        edit distance.
        >>> replacer = SpellingReplacer()
        >>> replacer.replace('cookbok')
        'cookbook'
        """
        def __init__(self, dict_name='en', max_dist=2):
                self.spell_dict = enchant.Dict(dict_name)
                self.max_dist = max_dist

        def replace(self, word):
                if self.spell_dict.check(word):
                        return word

                suggestions = self.spell_dict.suggest(word)

                if suggestions and edit_distance(word, suggestions[0]) <= self.max_dist:
                        return suggestions[0]
                else:
                        return word

class CustomSpellingReplacer(SpellingReplacer):
        """ SpellingReplacer that allows passing a custom enchant dictionary, such
        a DictWithPWL.
        >>> d = enchant.DictWithPWL('en_US', 'mywords.txt')
        >>> replacer = CustomSpellingReplacer(d)
        >>> replacer.replace('nltk')
        'nltk'
        """
        def __init__(self, spell_dict, max_dist=2):
                self.spell_dict = spell_dict
                self.max_dist = max_dist

#########################
## Replacing Synonyms ##
#########################

class WordReplacer(object):
        """ WordReplacer that replaces a given word with a word from the word_map,
        or if the word isn't found, returns the word as is.
        >>> replacer = WordReplacer({'bday': 'birthday'})
        >>> replacer.replace('bday')
        'birthday'
        >>> replacer.replace('happy')
```

```
        'happy'
        """
        def __init__(self, word_map):
                self.word_map = word_map

        def replace(self, word):
                return self.word_map.get(word, word)

class CsvWordReplacer(WordReplacer):
        """ WordReplacer that reads word mappings from a csv file.
        >>> replacer = CsvWordReplacer('synonyms.csv')
        >>> replacer.replace('bday')
        'birthday'
        >>> replacer.replace('happy')
        'happy'
        """
        def __init__(self, fname):
                word_map = {}

                for line in csv.reader(open(fname)):
                        word, syn = line
                        word_map[word] = syn

                super(CsvWordReplacer, self).__init__(word_map)

class YamlWordReplacer(WordReplacer):
        """ WordReplacer that reads word mappings from a yaml file.
        >>> replacer = YamlWordReplacer('synonyms.yaml')
        >>> replacer.replace('bday')
        'birthday'
        >>> replacer.replace('happy')
        'happy'
        """
        def __init__(self, fname):
                word_map = yaml.load(open(fname))
                super(YamlWordReplacer, self).__init__(word_map)

########################################
## Replacing Negations with Antonyms ##
########################################

class AntonymReplacer(object):
        def replace(self, word, pos=None):
                """ Returns the antonym of a word, but only if there is no ambiguity.
                >>> replacer = AntonymReplacer()
                >>> replacer.replace('good')
                >>> replacer.replace('uglify')
                'beautify'
                >>> replacer.replace('beautify')
                'uglify'
                """
                antonyms = set()

                for syn in wordnet.synsets(word, pos=pos):
```

```
                        for lemma in syn.lemmas:
                                for antonym in lemma.antonyms():
                                        antonyms.add(antonym.name)

                if len(antonyms) == 1:
                        return antonyms.pop()
                else:
                        return None

        def replace_negations(self, sent):
                """ Try to replace negations with antonyms in the tokenized sentence.
                >>> replacer = AntonymReplacer()
                >>> replacer.replace_negations(['do', 'not', 'uglify', 'our', 'code'])
                ['do', 'beautify', 'our', 'code']
                >>> replacer.replace_negations(['good', 'is', 'not', 'evil'])
                ['good', 'is', 'not', 'evil']
                """
                i, l = 0, len(sent)
                words = []

                while i < l:
                        word = sent[i]

                        if word == 'not' and i+1 < l:
                                ant = self.replace(sent[i+1])

                                if ant:
                                        words.append(ant)
                                        i += 2
                                        continue

                        words.append(word)
                        i += 1

                return words

class AntonymWordReplacer(WordReplacer, AntonymReplacer):
        """ AntonymReplacer that uses a custom mapping instead of WordNet.
        Order of inheritance is very important, this class would not work if
        AntonymReplacer comes before WordReplacer.
        >>> replacer = AntonymWordReplacer({'evil': 'good'})
        >>> replacer.replace_negations(['good', 'is', 'not', 'evil'])
        ['good', 'is', 'good']
        """
        pass

if __name__ == '__main__':
        import doctest
        doctest.testmod()

import nltk
from nltk.corpus import brown
import io, os
import logging
from replacers import RegexpReplacer
```

```
import re

#create output file
root = r'C:\PyCode'
outputfile = io.open(os.path.join(root,"bettertext2ndneg.txt"),'w')

# creating test set
root = r'C:\PyCode\data\neg' # root for test data
txt = "----------Beginning tagging process----------"
print txt

negfile = io.open(os.path.join(root,'NEGtweet.txt'),'r')
neglines = negfile.readlines()
contador = 1
for line in neglines:
    txt = "Sentence No.: %s"%contador
    print txt
    txt = "Original sentence: %s"%line
    print txt
    replacer = RegexpReplacer()
    outstep1 = replacer.replace(line)
    txt = "Replaced sentence: %s"%outstep1
    print txt
    outputfile.write(unicode(outstep1))
    contador = contador + 1
outputfile.flush()
outputfile.close()

import nltk
from nltk.corpus import brown
import io, os
import logging
from replacers import RegexpReplacer
import re

#create output file
root = r'C:\PyCode'
outputfile = io.open(os.path.join(root,"bettertext2ndpos.txt"),'w')

# creating test set
root = r'C:\PyCode\data\neg' # root for test data
txt = "----------Beginning tagging process----------"
print txt

negfile = io.open(os.path.join(root,'NEGtweet.txt'),'r')
neglines = negfile.readlines()
contador = 1
for line in neglines:
    txt = "Sentence No.: %s"%contador
    print txt
    txt = "Original sentence: %s"%line
    print txt
    replacer = RegexpReplacer()
    outstep1 = replacer.replace(line)
    txt = "Replaced sentence: %s"%outstep1
```

```
    print txt
    outputfile.write(unicode(outstep1))
    contador = contador + 1
outputfile.flush()
outputfile.close()

import nltk
from nltk.corpus import brown
import io, os
import logging
from replacers import RegexpReplacer
import re

#create output file
root = r'C:\PyCode'
outputfile = io.open(os.path.join(root,"outputtagged2ndneg.txt"),'w')

# training taggers
brown_tagged_sents=brown.tagged_sents(categories='news')
brown_sents=brown.sents(categories='news')
unigram_tagger = nltk.UnigramTagger(brown_tagged_sents)
brown_sents=brown.sents(categories='news')
unigram_tagger=nltk.UnigramTagger(brown_tagged_sents)

size = int(len(brown_tagged_sents))
print size
train_sents=brown_tagged_sents[:size]
unigram_tagger = nltk.UnigramTagger(train_sents)
bigram_tagger = nltk.BigramTagger(train_sents)
t0=nltk.DefaultTagger('NN')
t1=nltk.UnigramTagger(train_sents, backoff=t0)
t2=nltk.BigramTagger(train_sents, backoff=t1)
t2.evaluate(train_sents)

# creating test set
#root = r'C:\PyCode\data\pos' # root for test data
txt = "---------------------------Beginning tagging process----------------"
print txt

comillas = '"'
txtci = "[ "
txtcd = " ]"
negfile = io.open(os.path.join(root,'bettertext2ndneg.txt'),'r')
neglines = negfile.readlines()
contador = 1
outputfile.write(unicode(comillas))
outputfile.write(unicode(txtci))

for line in neglines:
    txt = "Sentence No. %s"%contador
    print txt
    txt = "Original sentence: %s"%line
    print txt
# using default tokenizer TreebankWordTokenizer
    words = nltk.word_tokenize(line)
```

```
    tagueado = t2.tag(words)
    txt = "Tagged sentence: %s"%tagueado
    print txt
    txt = "          "
    print txt
    outputfile.write(unicode(tagueado))
    contador = contador + 1

outputfile.write(unicode(txtcd))
outputfile.write(unicode(comillas))

outputfile.flush()
outputfile.close()
```

Appendix E

Python Code for Naïve Bayes (NB) & Maximum Entropy (ME) Classification Methods

The following Annex includes segments of code written in Python with the purpose of:

- Generating the classification of the test dataset using the Naïve Bayes (NB) algorithm
- Generating the classification of the test dataset using the Maximum Entropy (ME) algorithm

The code provided below, written in Python is available through the book 'Natural Language Processing with Python' by Bird [26] and the Perkins' book [173], 'Python Text Processing with NLTK 2.0 Cookbook' that provides interesting pieces of Python code oriented toward performing a variety of NLP tasks.

Note: this code has *not* be written by the author of this PhD Thesis, but written and distributed by the sources described in the previous paragraph. However, the modifications and adjustments required to generate the products required for our research are 100% mine. Below, the aforementioned code.

```
from nltk.corpus import movie_reviews
import io, os
import nltk
os.sys.path.append('C:\PyCode') # location of featx mudule
import featx
from nltk.classify.util import accuracy
#from featx import label_feats_from_corpus, split_label_feats
import collections
from nltk.corpus import stopwords, reuters
from nltk.collocations import BigramCollocationFinder
from nltk.metrics import BigramAssocMeasures
from nltk.probability import FreqDist, ConditionalFreqDist
import logging

#create output file
root = r'C:\PyCode'
outputfile = io.open(os.path.join(root,'outputnbnegsecond.txt'),'w')

print movie_reviews.categories()
# training classifier
lfeats = featx.label_feats_from_corpus(movie_reviews)
# split lfeats returning two lists: one with the first 75% of the
# words and the sencond 25%
train_feats, test_feats = featx.split_label_feats(lfeats) # default 0.75 (75%)
```

```
print len(train_feats) # lenght of the 75% of the list
print len(test_feats) # lenght of the 25% of the list

from nltk.classify import NaiveBayesClassifier
nb_classifier = NaiveBayesClassifier.train(train_feats)
print nb_classifier.labels()

# creating test set
root = r'C:\PyCode\data' # root for test data
#reader = nltk.corpus.reader.plaintext.PlaintextCorpusReader(root, '.*\.txt') # reader
txt = "----------Testing NEGATIVE file ----------"
print txt
# outputfile.write(unicode(txt + "\n"))

negfile = io.open(os.path.join(root,'neg\\NEGtweet.txt'),'r')
neglines = negfile.readlines()
contador = 0
# ========
for line in neglines:
    txt = "Processing sentence: %s"%line
    print txt
    outputfile.write(unicode(txt))
    words = nltk.word_tokenize(line) # using default tokenizer TreebankWordTokenizer
    negfeat = featx.bag_of_words(words)
    probs = nb_classifier.prob_classify(negfeat)
    txt = "Features classified as: %s with prob = %s"%(nb_classifier.classify(negfeat),
        probs.prob(nb_classifier.classify(negfeat)))
    print txt
    outputfile.write(unicode(txt + "\n\n"))
outputfile.flush()
outputfile.close()

from nltk.corpus import movie_reviews
import io, os
import nltk
os.sys.path.append('C:\PyCode') # location of featx mudule
import featx
from nltk.classify.util import accuracy
#from featx import label_feats_from_corpus, split_label_feats
import collections
from nltk.corpus import stopwords, reuters
from nltk.collocations import BigramCollocationFinder
from nltk.metrics import BigramAssocMeasures
from nltk.probability import FreqDist, ConditionalFreqDist
import logging

#create output file
root = r'C:\PyCode'
outputfile = io.open(os.path.join(root,'outputmenegsecond.txt'),'w')

print movie_reviews.categories()
# training classifier
lfeats = featx.label_feats_from_corpus(movie_reviews)
# split lfeats returning two lists: one with the first 75% of the
# words and the sencond 25%
```

```
train_feats, test_feats = featx.split_label_feats(lfeats) # default 0.75 (75%)
print len(train_feats) # lenght of the 75% of the list
print len(test_feats) # lenght of the 25% of the list

from nltk.classify import MaxentClassifier
me_classifier = MaxentClassifier.train(train_feats, algorithm='gis', trace=0,
    max_iter=3, min_lldelta=0.5)
print me_classifier.labels()

# creating test set
root = r'C:\PyCode\data' # root for test data
#reader = nltk.corpus.reader.plaintext.PlaintextCorpusReader(root, '.*\.txt') # reader
txt = "----------------------------Testing NEGATIVE file ----------------"
print txt
# outputfile.write(unicode(txt + "\n"))

negfile = io.open(os.path.join(root,'neg\\NEGtweet.txt'),'r')
neglines = negfile.readlines()
contador = 0
for line in neglines:
    contador = contador + 1
    lvacio = [] #initialise empty set
    elcero = 0
    txt = "Processing sentence: %s"%line
    print txt

    words = nltk.word_tokenize(line) # using default tokenizer TreebankWordTokenizer
    lvacio.append(contador)
    negfeat = featx.bag_of_words(words)

    probs = me_classifier.prob_classify(negfeat)
#     print words
    txt = "Features classified as: %s with prob = %s"%(me_classifier.classify(negfeat),
        probs.prob(me_classifier.classify(negfeat)))
    lvacio.append(probs.prob(me_classifier.classify(negfeat)))
    lvacio.append(me_classifier.classify(negfeat))
    lvacio.append(elcero)
#     print lvacio
    print txt
#     outputfile.write(unicode(txt + "\n\n"))
    outputfile.write(unicode(lvacio)) #write the list to file
outputfile.flush()
outputfile.close()
```

Appendix F

Samples of outputs of Syntactic Conversions Programs

The following Annex includes outputs of conversions of sentences in their initial formats to:

- List format
- PoS-tagged parse-tree
- Sub-sentences with replacement of lexicon terms available
- Final output to be analysed containing only relevant terms

The code utilised for these operations is part of the content provided in AnnexC.

F.1 Conversion process from string-raw data to analysis-ready data

1. Original Sentence:
("the rock is destined to be the 21st century's new conan and that he's going to make a splash even greater than arnold schwarzenegger , jean-claud van damme or steven segal . ")

2. List Lisp/Scheme Format Sentence:
(the rock is destined to be the 21st century's new conan and that he's going to make a splash even greater than arnold schwarzenegger , jean-claud van damme or steven segal .)

3. Converted into a tagged and parsed expression:
((the AT) (rock NN) (is BEZ) (destined VBN) (to TO) (be BE)
(the AT) (1st OD) (century NN) (new JJ) (conan NN)
(and CC) (that CS) (he PPS) (is BEZ) (going VBG) (to TO)
(make VB) (a AT) (splash NN) (even RB) (greater JJR)
(than CS) (arnold NN) (schwarzenegger NN) (jean NN)
(claud NN) (van NN) (damme NN) (or CC) (steven NN)
(segal NN) (punto punto))

Note: PoS labels, like AT, TO, VBG, etc., correspond to the PoS tags utilised in the code as per listed in Appendix C, Section C.5.

4. Existing lexicon terms replaced in sentence:
(rock is destined be century (#(new s 0.0 0.0 1.0 nocsor nomaxdist nomindist 1) pos) conan is going make

(#(splash v 0.0 0.0 1.0 nocsor nomaxdist nomindist 1 1) obj) (#(even r 0.125 0.0 0.875 nocsor nomaxdist nomindist 1 1) obj)
(#(greater a 0.5 0.25 0.25 nocsor nomaxdist nomindist 1 1) pos) arnold schwarzenegger jean claud van damme steven segal)

5. Non-contributing terms are eliminated from the expression:
((#(new s 0.0 0.0 1.0 nocsor nomaxdist nomindist 1 1) pos)
(#(splash v 0.0 0.0 1.0 nocsor nomaxdist nomindist 1 1) obj)
(#(even r 0.125 0.0 0.875 nocsor nomaxdist nomindist 1 1) obj)
(#(greater a 0.5 0.25 0.25 nocsor nomaxdist nomindist 1 1) pos))

Note: in an optimised version of the sentiment lexicon the attributes nocsor, nomaxdist, nomindist are deleted as described in Part IV, Chapter 17, Sub-section 17.1.3.

6. Non-contributing terms are eliminated from the expression (using updated Sentiment Lexicon structure in this instance):
((#(new s 0.0 0.0 1.0 1 1) pos)
(#(splash v 0.0 0.0 1.0 1 1) obj)
(#(even r 0.125 0.0 0.875 1 1) obj)
(#(greater a 0.5 0.25 0.25 1 1) pos))

Appendix G

Examples of the application of Semantic Rules & Negation

The following Annex includes samples of outputs once the following operations are applied to some example sentences:

- Semantic Rules
- Smart Negation

G.1 Semantic Rules - Examples

Example: It was a great night but the rain ruined it completely.

Sub-sentence 1: It was a great night.
Sub-sentence 2: the rain ruined completely.
Connecting Particle: **but**. Semantic rule $R11^{HSC}$ in Section 14.1.2, Table 14.1 will pick this sentence and will get rid of the first sub-sentence, producing:

Final sentence: the rain ruined it completely.

In turn this resulting sentence will be analysed by the HSC algorithm and will end up being assigned a negative connotation.

G.2 Smart Negation - Examples

A *replace* algorithm based on regular expressions contribute to breaking up particles like *doesn't, don't, aren't,* etc., into 'does not', 'do not' and 'are not', respectively (see Appendix D, Section D.2). Once the contraction related to negations are expanded, the smart negation algorithm produces a syntactic version of the sentence that reflects the scope of the negation, as explained in Chapter 14, Section 14.1.2.1.

Example: It was not a good movie.
The expanded version of the above sentence, including lexicon particles substitutions, would look like this:
((was VB) (not NOT) (a AT) (good ADJ) (movie NOUN))

Once simplified, it will turn into:

((was VB) (not NOT) (good ADJ)), and as the proper lexicon terms are replaced and *not* contributing terms are deleted, the following expression will be generated:
(#(good sa 0.9 0.1 0.0 1 1) pos) NEGATED)

The *negation management module* in HSC will realise that it is necessary to change the *polarity label* and *polarity scores* in the particles in scope of the negation, which in the aforementioned example corresponds to the word 'good'. Hence, the resulting expression is:
(#(good sa 0.1 0.9 0.0 1 1) neg)

Notice that now the 'polarity label = neg' and 'Positive Polarity Score = 0.1' and 'Negative Polarity Score = 0.9'. As a consequence, when the sentiment from this sentence is extracted, the particle 'good' will have a *polarity label* and a *Negative Polarity Score* carrying a negative connotation as as such, the sentence will be classified as having a *Negative Semantic Orientation*.